Synchronization of Bernoulli sequences on shared letters

Samy Abbes
Université Paris Diderot/IRIF CNRS UMR 8243
samy.abbes@univ-paris-diderot.fr

November 2016

Abstract

The topic of this paper is the distributed and incremental generation of long executions of concurrent systems, uniformly or more generally with weights associated to elementary actions.

Synchronizing sequences of letters on alphabets sharing letters are known to produce a trace in the concurrency theoretic sense, *i.e.*, a labeled partially ordered set. We study the probabilistic aspects by considering the synchronization of Bernoulli sequences of letters, under the light of Bernoulli and uniform measures recently introduced for trace monoids.

We introduce two algorithms that produce random traces, using only local random primitives. We thoroughly study some specific examples, the path model and the ring model, both of arbitrary size. For these models, we show how to generate any Bernoulli distributed random traces, which includes the case of uniform generation.

1—Introduction

The developments of concurrency theory and of model checking theory have urged the development of a theory of probabilistic concurrent systems. A central issue is the problem of the uniform generation of long executions of concurrent systems. Executions of a concurrent system can be represented as partial orders of events. Each partial order has several sequentializations, the combinatorics of which is non trivial. Therefore, the uniform generation of executions of a concurrent system is much different from the uniform generation of their sequentializations. The later can be done using uniform generation techniques for runs of transition systems, at the expense of an increasing amount of complexity due to the combinatorics of sequentializations [18, 9, 20]. Yet, it still misses the overall goal of uniform generation among executions of the system, since it focuses on their sequentializations.

We consider the framework of *trace monoids* [10], also called monoids with partial commutations [6]. Trace monoids have been studied as basic models of concurrent systems since several decades [17, 11]. One uses the algebraic commutation of generators of the monoid to render the concurrency of elementary actions. The elements of the trace monoid, called *traces*, represent the finite executions of the concurrent system [10]. More sophisticated concurrency models build on trace monoids, for instance executions of 1-safe Petri nets correspond to regular languages of trace monoids [19].

The topic of this paper is the effective uniform generation of large traces, *i.e.*, large elements in a trace monoid. It has several potential applications in the model checking and in the simulation of concurrent systems.

In a recent work co-authored with J. Mairesse [2], we have shown that the notion of Bernoulli measure for trace monoids provides an analogous, in a framework with concurrency, of classical Bernoulli sequences—*i.e.*, the mathematical model of memoryless coin tossing. In particular, Bernoulli measures encompass the *uniform measure*, an analogous for trace monoids of the maximal entropy measure. Therefore Bernoulli measures are an adequate theoretical ground to work with for the random generation of traces, in particular for the uniform generation.

Bernoulli sequences are highly efficiently approximated by random generators. An obvious, but nevertheless crucial feature of their generation is that it is incremental. For the random generation of large traces, we shall also insist that the generation procedure is incremental. Furthermore, another desirable feature is that it is distributed, in a sense that we explain now.

We consider trace monoids attached to networks of alphabets sharing common letters. The synchronization of several sequences of letters on different local alphabets sharing common letters is known to be entirely encoded by a unique element of a trace monoid. If Σ denotes the union of all local alphabets, then the *synchronization trace monoid* is the monoid with the presentation by generators and relations $\mathcal{M} = \langle \Sigma \mid ab = ba \rangle$, where (a, b) ranges over pairs of letters that do not occur in any common local alphabet. Hence, seeing local alphabets as "resources", two distinct letters a and b commute in $\mathcal{M}$ if and only if they do not share any common resource—a standard paradigm in concurrency theory.

In this framework, our problem rephrases as follows: given a synchronization trace monoid, design a probabilistic protocol to reconstruct a global random trace, uniformly among traces, and in a distributed and incremental way. By "distributed", we mean that the random primitives should only deal with the local alphabets. The expression "uniformly among traces" deserves also an explanation, since traces of a monoid are countably many. One interpretation is to fix a size k for target traces, and to retrieve a trace uniformly distributed among those of size k. Another interpretation is to consider infinite traces, *i.e.*, endless executions of the concurrent system. It amounts in an idealization of the case with large size traces. We then rely on the notion of uniform measure for infinite traces, which happens to have nicer properties than the uniform distribution on traces of fixed size. As explained above, the uniform measure belongs to the largest class of Bernoulli measures for trace monoids. Hence, a slightly more general problem is the distributed and incremental reconstruction of any Bernoulli measure attached to a synchronization trace monoid.

We introduce two algorithms that partly solve this problem, the Probabilistic Synchronization Algorithm (PSA) and the Probabilistic Full Synchronization Algorithm (PFSA), the later building on the former. According to the topology of the network, one algorithm or the other shall be applied. We show that the problem of generating traces according to a Bernoulli measure is entirely solved for some specific topologies of the network, namely for the path topology and for the ring topology—it could also be applied successfully to a tree topology. It is only partly solved for a general topology. Yet, even in the case of a general topology, our procedure outputs large random traces according to a Bernoulli scheme, although it is unclear how to tune the probabilistic parameters in order to obtain uniformity. Furthermore, the amount of time needed to obtain a trace of size k is linear with k in average.

Several works analyzing the exchange of information in concurrent systems restrict their studies to tree topologies, see for instance [13], and very few has been said on the probabilistic aspects. In particular, designing an incremental and distributed procedure to uniformly randomize a system with a ring topology has been an unsolved problem in the literature so far.

How does our method compare with standard generation methods based on tools from Analytic Combinatorics, such as Boltzmann samplers techniques? There exists

a normal form for traces, the so-called Cartier-Foata normal form, from which one derives a bijection between traces of a given trace monoid and words of a regular language. This seems to draw a direct connection with Boltzmann sampling of words from regular languages ([12, p.590], [4]). This approach however suffers from some drawbacks. First, the rejection mechanism which is at the very heart of the Boltzmann sampling approach prevents the construction to be incremental, as we seek. This could be avoided by considering instead the generation of the Markov chain of the elements of the normal form associated to the uniform measure on infinite traces, as investigated partly in [3]. But this leads to a second problem, namely that the randomness now concentrates on the set of cliques of the trace monoid, a set which size grows exponentially fast with the number of generators of the monoid in general. It henceforth misses the point of being a distributed generation.

Outline. Section 2 illustrates on small examples the two algorithms for generating random traces from a network of alphabets, the PSA and the PFSA, to be fully analyzed in forthcoming sections. Sections 3 and Section 4 are two preliminary sections, gathering material on the combinatorics of trace monoids for the first one, and material on Bernoulli measures for trace monoids for the second one. The non-random algorithms for the synchronization of sequences, possibly infinite, that we present in Section 3 seem to be new, although the fundamental ideas on which they rest are not new. A new result on the theory of Möbius valuations is given at the end of Section 4.

Our main contributions are organized in Sections 5 and 6. Section 5 is devoted to the Probabilistic Synchronization Algorithm (PSA). The analysis of the algorithm itself is quite simple and short. The most striking contributions are in the examples. In particular, for the surprising case of the path model, the PSA is shown to work the best one could expect.

Section 6 is devoted to the description and to the analysis of the Probabilistic Full Synchronization Algorithm (PFSA), both from the probabilistic point of view and from the complexity point of view. Examples are examined; the ring model of arbitrary size is precisely studied, and we obtain the satisfying result of simulating any Bernoulli scheme on infinite traces by means of the PFSA.

Finally, Section 7 discusses some additional complexity issues and suggests perspectives.

2—Illustrating the PSA and PFSA algorithms

In this section, we illustrate our two generation algorithms on small examples, as well as the tuning of their probabilistic parameters, at an informal and descriptive level.

2.1 — Illustrating the Probabilistic Synchronization Algorithm (1)

Let a_0, a_1, a_2, a_3, a_4 be five distinct symbols, and consider the four alphabets:

$$\Sigma_1 = \{a_0, a_1\}, \qquad \Sigma_2 = \{a_1, a_2\}, \qquad \Sigma_3 = \{a_2, a_3\}, \qquad \Sigma_4 = \{a_3, a_4\}.$$

To each alphabet Σ_i is attached a device able to produce a random sequence $Y_i = (Y_{i,1}, Y_{i,2}, \dots)$ of letters $Y_{i,j} \in \Sigma_i$. The random letters are independent and identically distributed, according to a distribution $p_i = \begin{pmatrix} p_i(a_{i-1}) & p_i(a_i) \end{pmatrix}$ to be determined later, but with positive coefficients. Furthermore, we assume that the devices themselves are probabilistically independent with respect to each other. For instance, the beginnings of the four sequences might be:

$$Y_1 = (a_1 a_0 a_0 a_1 a_0 \dots) \qquad Y_2 = (a_1 a_2 a_2 a_2 a_2 \dots)$$
$$Y_3 = (a_2 a_2 a_3 a_3 a_2 \dots) \qquad Y_4 = (a_4 a_3 a_3 a_3 a_4 \dots)$$

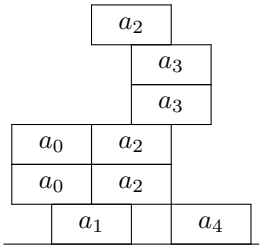


Figure 1: Heap of pieces corresponding to the vector Y

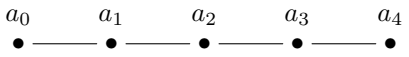


Figure 2: Synchronization graph of the path model with five generators

Then we stack these four sequences in a unique vector Y with four coordinates, and we join the matching letters from one coordinate to its neighbor coordinates, in their order of appearance. The border elements a_0 and a_4 do not match with any other element; some elements are not yet matched (for example, the second occurrence of a_1 in Y_1), and we do not picture them. This yields:

$$Y = \begin{pmatrix} a_1 & a_0 & a_0 & & & & \dots \\ | & & & & & & \\ a_1 & a_2 & a_2 & & & a_2 & \dots \\ & | & | & & & | & \\ & a_2 & a_2 & a_3 & a_3 & a_2 & \dots \\ & & & | & | & & \\ a_4 & & & a_3 & a_3 & & \dots \end{pmatrix}$$

Finally, we identify each matching pair with a single *piece*, labeled with the matching letter and we impose a rotation of the whole picture by a quarter turn counterclockwise. We obtain the *heap of pieces* depicted in Figure 1. It is apparent on this picture that pieces a_i and a_{i+1} share a sort of common resource. Therefore we depict the topology of the system resulting from the synchronization of the network of alphabets $(\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4)$ in Figure 2, by drawing an undirected graph with pieces as vertices, and an edge between two pieces whenever they share a common resource, or equivalently, whenever they both appear in the same alphabet.

The PSA algorithm consists, for each device attached to each alphabet Σ_i, to do the following: 1) Generate its own local random sequence; 2) Communicate with the neighbors in order to tag the matching of each of the occurring letters.

Since the coefficients $p_i(a_{i-1})$ and $p_i(a_i)$ are positive, each coordinate Y_i has infinitely many occurrences of both letters a_{i-1} and a_i. Furthermore, each occurrence of a letter a_j with $j \neq 0, 4$ will eventually match a corresponding occurrence in a neighbor coordinate. Therefore the random heap that we obtain, say ξ, is infinite if the algorithm keeps running forever. The PSA produces incremental finite approximations of ξ. The probabilistic analysis of the PSA consists in determining the probability distribution of the theoretical infinite random heap ξ that the PSA would produce if it was to run indefinitely.

In order to characterize the probability distribution of ξ, we shall denote by $\uparrow x$, for each finite possible heap x, the probabilistic event that ξ *starts with* x. This means

that the finite heap x can be seen at the bottom of ξ. Equivalently: $\xi \in \uparrow x$ if, after waiting long enough, the finite heap x can be seen at the bottom of the current heap produced by the PSA.

We aim at evaluating the probability $\mathbb{P}(\uparrow x)$ for every finite heap x. For this, consider an arbitrary possible sequentialization of x, seen as a successive piling of different occurrences of the elementary pieces, which we write symbolically as $x = x_1 \cdot \ldots \cdot x_k$ with $x_j \in \{a_0, a_1, a_2, a_3, a_4\}$. Then $\mathbb{P}(\uparrow x)$ is obtained as the following product:

$$\mathbb{P}(\uparrow x) = t_{x_1} \cdot \ldots \cdot t_{x_k},$$

where the coefficient t_{x_j} corresponds to the probability of having the letter x_j appearing either on its single coordinate if $x_j = a_0$ or $x_j = a_4$, or on both coordinates to which it belongs otherwise. Hence:

$$t_{a_0} = p_1(a_0) \qquad t_{a_1} = p_1(a_1) \cdot p_2(a_1) \qquad t_{a_2} = p_2(a_2) \cdot p_3(a_2)$$
$$t_{a_3} = p_3(a_3) \cdot p_4(a_3) \qquad t_{a_4} = p_4(a_4).$$

For instance, if the local probabilities p_i were uniform, $p_i = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix}$, we would have:

$$t_{a_0} = \frac{1}{2} \qquad t_{a_1} = \frac{1}{4} \qquad t_{a_2} = \frac{1}{4} \qquad t_{a_3} = \frac{1}{4} \qquad t_{a_4} = \frac{1}{2}.$$

This choice, which may seem natural at first, would actually produce a random heap with a bias, namely it would favor the appearance of pieces a_0 and a_4. Since $t_{a_0} = 1/2$ and $t_{a_1} = 1/4$ in this case, it would produce on average twice more occurrences of a_0 than occurrences of a_1.

However, for uniform generation purposes, it is desirable to obtain all coefficients t_{a_i} *equal*. Can we tune the initial probability distributions p_i in order to achieve this result? Introduce the Möbius polynomial $\mu(z) = 1 - 5z + 6z^2 - z^3$, and consider its root of smallest modulus, namely $q_0 \approx 0.308$. It turns out that the only way to have all coefficients t_{a_i} equal is to make them precisely equal to q_0. In turn, this imposes conditions on the local probability distributions p_1, p_2, p_3, p_4 with only one solution, yielding for this example:

$$\begin{array}{ccccc}
p_1(a_0) = q_0 \approx 0.308 & & \xrightarrow{\cdot+\cdot=1} & & p_1(a_1) = 1 - q_0 \approx 0.692 \\
& \searrow^{\cdot\times\cdot=q_0} & & \nearrow_{\cdot\times\cdot=q_0} & \\
p_2(a_1) = \frac{q_0}{1-q_0} \approx 0.445 & & \xrightarrow{\cdot+\cdot=1} & & p_2(a_2) = \frac{1-2q_0}{1-q_0} \approx 0.555 \\
& \searrow^{\cdot\times\cdot=q_0} & & \nearrow_{\cdot\times\cdot=q_0} & \\
p_3(a_2) = \frac{1-2q_0}{1-q_0} \approx 0.555 & & \xrightarrow{\cdot+\cdot=1} & & p_3(a_3) = \frac{q_0}{1-q_0} \approx 0.445 \\
& \searrow^{\cdot\times\cdot=q_0} & & \nearrow_{\cdot\times\cdot=q_0} & \\
p_4(a_3) = 1 - q_0 \approx 0.692 & & \xrightarrow{\cdot+\cdot=1} & & p_4(a_4) = q_0 \approx 0.308
\end{array}$$

This array of positive numbers has the sought property that the product of any two numbers along the depicted diagonals equals q_0, and the sum of every line equals 1; and q_0 is the only real allowing this property for an array of this size. Running the PSA with these values for the local probability distributions produces a growing random heap which is uniform, in a precise meaning that will be formalized later in the paper.

2.2 — Illustrating the Probabilistic Synchronization Algorithm (2)

Consider the following network of alphabets with the four letters a_0, a_1, a_2, a_3:

$$\Sigma_1 = \{a_0, a_1\}, \qquad \Sigma_2 = \{a_1, a_2\}, \qquad \Sigma_3 = \{a_2, a_3\}, \qquad \Sigma_4 = \{a_3, a_0\}.$$

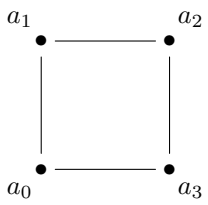


Figure 3: Synchronization graph of the ring model with four generators

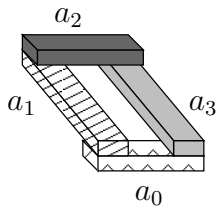


Figure 4: Output heap resulting from an instance of the PSA for the ring model with four generators

Analogously to the previous example, the synchronization graph for this network is depicted in Figure 3. Let us try to apply the same generation technique as in the previous example. For each alphabet Σ_i, we consider a random sequence Y_i of letters of this alphabet. Given the symmetry of the network of alphabets, in order to obtain a uniform distribution we need to consider this time uniform local distributions $p_i = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix}$ for $i = 1, 2, 3, 4$. This yields for instance:

$$Y_1 = a_0a_1a_1a_0\ldots \quad Y_2 = a_1a_2a_2a_1\ldots \quad Y_3 = a_3a_2a_3a_2\ldots \quad Y_4 = a_0a_3a_0a_3\ldots$$

The construction of the stacking vector Y with the matching of letters yields:

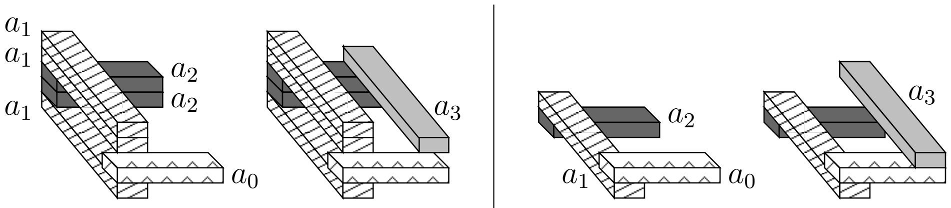
$$Y = \begin{pmatrix} a_0 & a_1 & \ldots & \\ & a_1 & a_2 & \ldots \\ & a_3 & a_2 & \ldots \\ a_0 & a_3 & \ldots & \end{pmatrix}$$

the two occurrences of a_0 being connected.

The corresponding heap is depicted on Figure 4. It is then impossible to extend this heap while still respecting the sequences Y_1, Y_2, Y_3, Y_4; not because some letter is waiting for its matching, but because the different letters already present are blocking each other, creating a cycle that prevents to interpret the rest of the vector as an extension of the heap of Figure 4. Such a cycle will appear with probability 1 at some point, whatever the sequences Y_1, Y_2, Y_3, Y_4. Henceforth the PSA only outputs finite heaps for this model.

2.3 — Illustrating the Probabilistic Full Synchronization Algorithm

In the previous example, we have seen that the PSA outputs a finite heap with probability 1, whereas we would like to obtain heaps arbitrary large. The Probabilistic Full Synchronization Algorithm (PFSA) is designed to solve this issue. We will now describe how to tune it, for the ring model with four generators introduced above and illustrated in Figure 3, in order to produce infinite heaps uniformly distributed.



(i) ξ_1' (ii) reject (iii) ξ_1' (iv) accept

Figure 5: First iteration of the PFSA. (i) The finite output ξ_1' of the PSA on three generators a_0, a_1, a_2. (ii) Adding the piece a_3 on top of ξ_1', the obtained heap is not pyramidal since occurrences of a_1 can be removed without moving a_3. (iii) Another instance of the heap ξ_1' on three generators a_0, a_1, a_2. (iv) This time, the new heap ξ_1 obtained by adding a_3 on top of ξ_1' is pyramidal.

First, the theory of Bernoulli measures for trace monoids (see below in Section 4) tells us that each piece of the monoid must be given the probabilistic weight q_1, root of smallest modulus of the Möbius polynomial $\mu(z) = 1 - 4z + 2z^2$, hence $q_1 = 1 - \sqrt{2}/2$.

Second, we arbitrarily choose a piece, say a_3, to be removed. We are left with the following network of alphabets: $\Sigma_1' = \{a_0, a_1\}$, $\Sigma_2' = \{a_1, a_2\}$, $\Sigma_3' = \{a_2\}$, $\Sigma_4' = \{a_0\}$. From the heaps point of view, this is equivalent to the network with only two alphabets (Σ_1', Σ_2'), and sharing thus the only letter a_1. We will now design a variant of the PSA algorithm, to be applied to the network (Σ_1', Σ_2'), outputting *finite* heaps with probability 1, and attributing the probabilistic parameter q_1 to each of the three pieces a_0, a_1, a_2. This is possible by considering *sub-probability* distributions p_1', p_2' associated to Σ_1', Σ_2' instead of *probability* distributions for the local generations. A possible choice, although not the unique one, is the following:

$$p_1'(a_0) = q_1 = 1 - \sqrt{2}/2 \quad \xrightarrow{\cdot + \cdot = 1} \quad p_1'(a_1) = 1 - q_1 = \sqrt{2}/2$$

$$\cdot \times \cdot = q_1$$

$$p_2'(a_1) = \frac{q_1}{1-q_1} = \sqrt{2} - 1 \quad \xrightarrow{\cdot + \cdot < 1} \quad p_2'(a_2) = q_1 = 1 - \sqrt{2}/2$$

Note that the first line sums up to 1 whereas the second line sums up to less than 1, and this guaranties that the PSA executed with these values will output a finite heap and stop with probability 1.

Let ξ_1' be the output of the PSA on (Σ_1', Σ_2') with the above parameters. It is a finite heap built with occurrences of a_0, a_1 and a_2 only. Then, we add the piece a_3 on top of ξ_1', and we check whether the obtained heap is *pyramidal*, which means that no piece can be removed from it without moving the last piece a_3 (see Figure 5). If it is not pyramidal, we reject it, and re-run the PSA, producing another instance of ξ_1', until $\xi_1' \cdot a_3$ is pyramidal. At the end of this process, we obtain a pyramidal heap $\xi_1 = \xi_1' \cdot a_3$ built with the four generators a_0, a_1, a_2, a_3, and a unique occurrence of a_3.

The process of computing this heap ξ_1 is the first loop of the PFSA. We initialize what will be the final output of the algorithm by setting $y_1 = \xi_1$. We then reproduce the above random procedure, yielding a pyramidal heap ξ_2 and then we form the heap $y_2 = y_1 \cdot \xi_2$, obtained by simply piling up ξ_2 on top of y_1. We iterate this procedure: outputting ξ_3 pyramidal, we put $y_3 = y_2 \cdot \xi_3$, and so on. Then the increasing heaps $y_1, y_2, \ldots$ approximate an infinite heap on the four generators a_0, a_1, a_2, a_3 which we claim to be uniformly distributed.

3—Preliminaries on trace monoids

An *alphabet* is a finite set, usually denoted by Σ, the elements of which are called *letters*. The free monoid generated by Σ is denoted by Σ^*.

3.1 — Combinatorics of trace monoids

3.1.1 *Definitions* — An *independence pair* on the alphabet Σ is a binary, irreflexive and symmetric relation on Σ, denoted I. The *trace monoid* $\mathcal{M} = \mathcal{M}(\Sigma, I)$ is the presented monoid $\mathcal{M} = \langle \Sigma \mid ab = ba \text{ for all } (a,b) \in I \rangle$. Hence, if $\mathcal{R}$ is the smallest congruence on Σ^* that contains all pairs (ab, ba) for (a,b) ranging over I, then $\mathcal{M}$ is the quotient monoid $\mathcal{M} = \Sigma^*/\mathcal{R}$. Elements of $\mathcal{M}$ are called *traces* [10]. The unit element is denoted by $\mathbf{e}$, and the concatenation in $\mathcal{M}$ is denoted by "$\cdot$".

The *length* $|x|$ of a trace $x \in \mathcal{M}$ is the length of any word in the equivalence class x. The left divisibility relation on $\mathcal{M}$ is denoted by $\leq$, it is defined by $x \leq y \iff \exists z \in \mathcal{M} \quad y = x \cdot z$.

3.1.2 *Cliques* — A *clique* of $\mathcal{M}$ is any element $x \in \mathcal{M}$ of the form $x = a_1 \cdot \ldots \cdot a_n$ with $a_i \in \Sigma$ and such that $(a_i, a_j) \in I$ for all $i \neq j$. Cliques thus defined are in bijection with the cliques, in the graph-theoretic sense, of the pair (Σ, I) seen as an undirected graph, that is to say, with the set of complete sub-graphs of (Σ, I).

The set of cliques is denoted by $\mathscr{C}$. The unit element is a clique, called the empty clique. The set of non empty cliques is denoted by $\mathfrak{C}$.

For instance, for $\mathcal{M} = \langle a_0, a_1, a_2, a_3, a_4 \mid a_i a_j = a_j a_i \text{ for } |i-j| > 1 \rangle$, we have $\mathscr{C} = \{\mathbf{e},\ a_0,\ a_1,\ a_2,\ a_3,\ a_4,\ a_0a_2,\ a_0a_3,\ a_0a_4,\ a_1a_3,\ a_1a_4,\ a_2a_4,\ a_0a_2a_4\}$. We call this trace monoid the *path model with five generators*, it corresponds to the example introduced in Section 2.1. Note that the synchronization graph depicted in Figure 2 is not (Σ, I) but its complementary.

3.1.3 *Growth series and Möbius polynomial* — The *growth series* of $\mathcal{M}$ is the formal series

$$Z_{\mathcal{M}}(t) = \sum_{x \in \mathcal{M}} t^{|x|}\,.$$

The *Möbius polynomial* [6] is the polynomial $\mu_{\mathcal{M}}(t)$ defined by

$$\mu_{\mathcal{M}}(t) = \sum_{c \in \mathscr{C}} (-1)^{|c|} t^{|c|}\,.$$

For the path model with five generators introduced above, we have $\mu_{\mathcal{M}}(t) = 1 - 5t + 6t^2 - t^3$.

The Möbius polynomial is the formal inverse of the growth series [6, 22]:

$$Z_{\mathcal{M}}(t) = 1/\mu_{\mathcal{M}}(t)\,.$$

The Möbius polynomial has a unique root of smallest modulus, say p_0. This root is real and lies in $(0, 1]$, and coincides with the radius of convergence of the power series $Z_{\mathcal{M}}(t)$ [16, 14].

3.1.4 *Multivariate Möbius polynomial* — The Möbius polynomial has a multivariate version, $\mu_{\mathcal{M}}(t_1, \ldots, t_N)$ where $t_1, \ldots, t_N$ are formal variables associated with the generators $a_1, \ldots, a_N$ of $\mathcal{M}$. It is defined by:

$$\mu_{\mathcal{M}}(t_1, \ldots, t_N) = \sum_{c \in \mathscr{C}} (-1)^{|c|} t_{j_1} \cdot \ldots \cdot t_{j_c}\,,$$

where the variables $t_{j_1}, \ldots, t_{j_c}$ correspond to the letters such that $c = a_{j_1} \cdot \ldots \cdot a_{j_c}$.

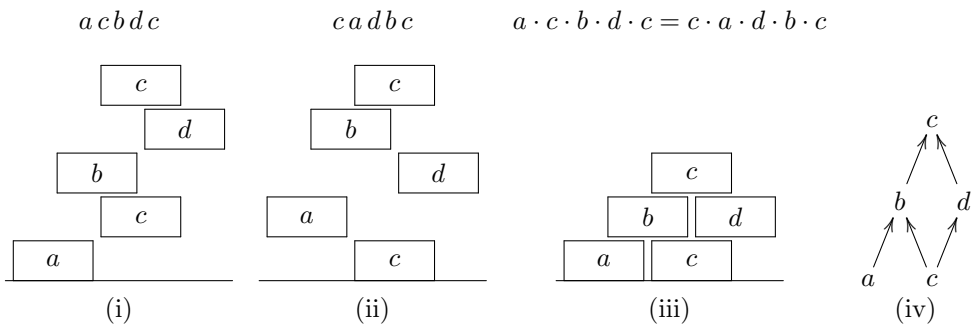


Figure 6: (i)–(ii): representation of the two congruent words *acbdc* and *cadbc* in $\langle a, b, c, d \mid ac = ca,\ ad = da,\ bd = db\rangle$. (iii): the resulting trace $a \cdot c \cdot b \cdot d \cdot c$, represented as a heap of pieces. (iv): the associated labeled ordered poset. For instance $a < d$ does not hold since $acbdc \to cabdc \to cadbc \to cdabc$.

3.1.5 *Irreducibility* — The *dependence relation* associated with $\mathcal{M} = \mathcal{M}(\Sigma, I)$ is the binary, symmetric and reflexive relation $D = (\Sigma \times \Sigma) \setminus I$. The trace monoid $\mathcal{M}$ is *irreducible* if the pair (Σ, D) is connected as a non oriented graph.

If $\mathcal{M}$ is irreducible, then the root p_0 of $\mathcal{M}$ is simple [16].

3.2 — The heap of pieces interpretation of traces

3.2.1 *The picture* — Viennot's theory provides a visualization of traces as *heaps of pieces* [22]. Picture each trace as the piling of dominoes labeled by the letters of the alphabet, and such that dominoes associated with two letters a and b fall to the ground according to parallel lanes, and disjoint if and only if $(a, b) \in I$. See an illustration on Figure 6, (i)–(iii), for $\mathcal{M} = \langle a, b, c, d \mid ac = ca,\ ad = da,\ bd = db\rangle$.

3.2.2 *Traces as labeled ordered sets* — The formalization of this picture is done by interpreting each trace x of a trace monoid as an equivalence class, up to isomorphism, of a Σ-labeled partial order, which we describe now.

Let x be a trace, written as a product of letters $x = a_1 \cdot \ldots \cdot a_n$. Let $\overline{x} = \{1, \ldots, n\}$, and define a labeling $\phi : \overline{x} \to \Sigma$ by $\phi(i) = a_i$. Equip $\overline{x}$ with the natural ordering on integers $\leq$. Then remove the pair (i, j) from the strict ordering relation $<$ on $\overline{x}$ whenever, by a finite number of adjacent commutations of distinct letters, the sequence $(a_1, \ldots, a_n)$ can be transformed into a sequence $(a_{\sigma(1)}, \ldots, a_{\sigma(n)})$ such that $\sigma(i) > \sigma(j)$.

The set of remaining ordered pairs (i, j) is a partial ordering on $\overline{x}$, which only depends on x, and which defines the heap associated with x. See Figure 6, (iv).

3.3 — Completing trace monoids with infinite traces

3.3.1 *Infinite traces* — Let $(x_n)_{n\geq 1}$ and $(y_n)_{n\geq 1}$ be two nondecreasing sequences of traces: $x_n \leq x_{n+1}$ and $y_n \leq y_{n+1}$ for all integers $n \geq 1$. We identify $(x_n)_{n\geq 1}$ and $(y_n)_{n\geq 1}$ whenever they satisfy:

$$(\forall n \geq 1 \quad \exists m \geq 1 \quad x_n \leq y_m) \wedge (\forall n \geq 1 \quad \exists m \geq 1 \quad y_n \leq x_m)\,.$$

This identification is an equivalence relation between nondecreasing sequences. The quotient set is denoted by $\overline{\mathcal{M}}$. It is naturally equipped with a partial ordering, such that the mapping $\mathcal{M} \to \overline{\mathcal{M}}$ associating the (equivalence class of the) constant sequence $x_n = x$ to any trace $x \in \mathcal{M}$, is an embedding of partial orders.

We identify thus $\mathcal{M}$ with its image in $\overline{\mathcal{M}}$ through the embedding $\mathcal{M} \to \overline{\mathcal{M}}$. The set $\partial\mathcal{M}$ of *infinite traces* is defined by:

$$\partial\mathcal{M} = \overline{\mathcal{M}} \setminus \mathcal{M}\,.$$

The set $\partial\mathcal{M}$ is called the *boundary* of $\mathcal{M}$ [2]. Visually, elements of $\partial\mathcal{M}$ correspond to *infinite countable heaps*, that is to say, limiting heaps obtained by piling up countable many pieces.

3.3.2 *Properties of* $(\overline{\mathcal{M}}, \leq)$ — The partially ordered set $(\overline{\mathcal{M}}, \leq)$ is complete with respect to least upper bound of nondecreasing sequences. And any element $\xi \in \overline{\mathcal{M}}$ is the least upper bound of a nondecreasing sequence of elements of $\mathcal{M}$ [2].

3.4 — Synchronization of sequences

In this section we introduce an alternative way of looking at trace monoids, by means of vectors of words. This emphasizes the distributed point of view on trace monoids.

3.4.1 *Network of alphabets and synchronizing trace monoid.* — A family $(\Sigma_1, \ldots, \Sigma_N)$ of N alphabets, not necessarily disjoint, is called a *network of alphabets*. Let $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_N$. Define $R = \{1, \ldots, N\}$ as the set of *resources*, and consider the product monoid

$$\mathcal{H} = (\Sigma_1)^* \times \ldots \times (\Sigma_N)^*\,. \tag{1}$$

Identify each $a \in \Sigma$ with the element $H(a) = (a_i)_{i \in R}$ of $\mathcal{H}$ defined by:

$$\forall i \in R \quad a_i = \begin{cases} \epsilon \text{ (the empty word)}, & \text{if } a \notin \Sigma_i\,, \\ a, & \text{if } a \in \Sigma_i\,. \end{cases}$$

For each letter $a \in \Sigma$, the *set of resources* associated with a is the subset $R(a)$ defined by

$$R(a) = \{i \in R \ : \ a \in \Sigma_i\}\,.$$

Let $\mathscr{G}$ be the sub-monoid of $\mathcal{H}$ generated by the collection $\{H(a) \ : \ a \in \Sigma\}$. Then $\mathscr{G}$ is isomorphic with the trace monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$, where the independence relation I on Σ is defined by

$$(a, b) \in I \iff R(a) \cap R(b) = \emptyset. \tag{2}$$

The monoid $\mathcal{M}$ is called the *synchronization trace monoid*, or simply the *synchronization monoid*, of the network $(\Sigma_1, \ldots, \Sigma_N)$. Examples have been given in Section 2. It can be proved that every trace monoid is isomorphic to a synchronization trace monoid [7].

3.4.2 *Synchronization of sequences.* — Let $(\Sigma_1, \ldots, \Sigma_N)$ be a network of alphabets. Each monoid $(\Sigma_i)^*$ being equipped with the prefix ordering, we equip the product monoid $\mathcal{H}$ defined in (1) with the product order, and the sub-monoid $\mathscr{G}$ with the induced order.

Assume given a vector of sequences $Y = (Y_1, \ldots, Y_N) \in \mathcal{H}$. The *synchronization trace* of Y is defined as the following least upper bound in $\mathscr{G}$:

$$X = \bigvee \{Z \in \mathscr{G} \ : \ Z \leq Y\}. \tag{3}$$

This least upper bound does indeed exist in $\mathscr{G}$: it is obtained by taking least upper bounds component-wise, which are all well defined.

Identifying $\mathscr{G}$ with the trace monoid $\mathcal{M}$ defined in (2) and above, the element X is thus the largest *trace* among those tuples below Y.

For example, consider the ring model with four generators introduced in Section 2.2, with $\Sigma_1 = \{a_0, a_1\}$, $\Sigma_2 = \{a_1, a_2\}$, $\Sigma_3 = \{a_2, a_3\}$ and $\Sigma_4 = \{a_3, a_0\}$, and the vector:

$$Y = \begin{pmatrix} a_0a_1a_1a_0 \\ a_1a_2a_2a_1 \\ a_3a_2a_3a_2 \\ a_0a_3a_0a_3 \end{pmatrix}$$

One easily convinces oneself that the synchronization of Y is the following:

$$X = \begin{pmatrix} a_0a_1 \\ a_1a_2 \\ a_3a_2 \\ a_0a_3 \end{pmatrix} \quad \text{corresponding to } a_0 \cdot a_1 \cdot a_3 \cdot a_2 \text{ in the trace monoid.}$$

We shall see below an algorithmic way to determine the synchronization of any given vector $Y \in \mathcal{H}$.

3.4.3 *On-line computation features.* — The algorithmic computation of the synchronization trace of a given vector of sequences might belong to the folklore of concurrency theory, see for instance [7]. In what follows, an special emphasis is given to the distributed and on-line nature of this computation.

The remaining of this section is devoted to provide an algorithm taking a vector of sequences $Y \in \mathcal{H}$ as input, and outputting the synchronization trace of Y. Furthermore, the input trace Y might not be entirely known at the time of computation. Instead, we assume that only a sub-trace $Y' \leq Y$ feeds the algorithm, together with the knowledge whether some further input is about to come or not. Therefore, we need our algorithm to produce the *two* following outputs:

1. The best approximation X' of the synchronization trace X, given only the input Y', which is merely the synchronization trace of Y'.
2. And one of the following tags:
   - DL for "Deadlock" if some more input is about to come, and yet the algorithm has already reached the synchronization trace of Y, *i.e.*, if $X' = X$ whatever the continuation of Y'.
   - WFI for "Waiting for Input" if some more input is about to come, and the trace X' might be extended.
   - EOF for "End of file" if there is no more input to come.

3.4.4 *Detecting deadlocks* — Consider, as in Sections 3.4.1—3.4.2, a network $(\Sigma_1, \ldots, \Sigma_N)$ of alphabets. Given a finite word $Y_i \in (\Sigma_i)^*$, we denote by $\overline{Y}_i$ the word $\overline{Y}_i = Y_i \cdot \dagger_i$, where $\dagger_i$ is an additional symbol which can be either EOF of WFI, to be interpreted as "no more input will ever come" if $\dagger_i = \text{EOF}$, or as "some more input might arrive" if $\dagger_i = \text{WFI}$. Finally, let $\overline{Y} = (\overline{Y}_1, \ldots, \overline{Y}_N)$.

We first introduce a basic routine described in pseudo-code below (Algorithm 1). The input of the routine is the vector $\overline{Y}$. Its output is as follows:

1. If the synchronization trace X of Y is non empty, then the routine outputs a minimal piece of X.
2. If the synchronization trace X of Y is empty, then the routine outputs a flag ‡ explaining why X is empty:

a) If $\overline{Y} = (\text{EOF}, \ldots, \text{EOF})$, then $\ddagger = \text{EOF}$. Note that this necessarily entails that Y is empty.

b) In case at least one input component of $\overline{Y}$ does not carry the symbol $\dagger_i = \text{EOF}$, then:

   i. If adding some letters to some of the components of Y carrying $\dagger_i = \text{WFI}$ could yield a non empty synchronization trace, then $\ddagger = \text{WFI}$.

   ii. If no letter can be added to the components of Y carrying $\dagger_i = \text{WFI}$ to yield a non empty synchronization trace, then $\ddagger = \text{DL}$.

To detect if a piece u is minimal in a heap, when the heap is given by its representation in $\mathscr{G}$, it is necessary and sufficient to check that it is minimal in all the coordinates where it occurs, that is to say, in all the components belonging to the set of resources $\mathcal{R}(u)$. This justifies that Algorithm 1 is sound. Observe that the algorithm is non deterministic, as there may be several minimal pieces.

Algorithm 1 executes in constant time, the constant growing linearly with the number of edges in the dependence graph (Σ, D) of the associated trace monoid. If communicating processes are devised, one per each alphabet, and able to write to a common register, they will be able to perform Algorithm 1 in a distributed way. We shall not work out the details of the distributed implementation, since it would be both outside the scope of the paper and out of the range of expertise of the author. People from the distributed algorithms community will probably find it routine.

3.4.5 *Computation of the synchronization trace.* — With Algorithm 1 at hand, we can now give an algorithm to compute the synchronization trace of a given vector of sequences $Y \in \mathcal{H}$. This is the topic of the Synchronization Algorithm (Algorithm 2), of which we give the pseudo-code below, and which takes as input a vector $\overline{Y} = (\overline{Y}_1, \ldots, \overline{Y}_N)$ of the same kind as Algorithm 1.

The Synchronization Algorithm iteratively executes Algorithm 1, and collects the minimal pieces thus obtained to form it own output. If $Y = (Y_1, \ldots, Y_N) \in \mathcal{H}$ and if $M \leq Y$, then we denote by $Z = M \backslash Y$ the unique vector $Z \in \mathcal{H}$ such that $M \cdot Z = Y$.

As in the requirements stated above in Section 3.4.3, the Synchronization Algorithm outputs both the synchronization trace of its input and a tag advertising if the computation is over, either because a deadlock has been reached or because the input feed is over, or if it is still waiting for some input that might cause the synchronization trace computed so far to be extended. This feature will be crucial when considering its execution on sequences of letters which are possibly infinite, see below in Section 3.4.6.

The Synchronization Algorithm can be executed in a distributed way by N communicating processes, one for each coordinate, and it runs in time linear with the size of the synchronization trace X.

3.4.6 *Feeding the Synchronization Algorithm with a possibly infinite input* — We have defined the synchronization trace of a vector $Y \in \mathcal{H}$ in Section 3.4.2. The same definition applies if one or several components of Y are infinite, by putting:

$$X = \bigvee \{Z \in \mathscr{G} \ : \ Z \leq Y\}, \tag{4}$$

and this least upper bound is always well defined in $\overline{\mathcal{M}}$ since least upper bounds of nondecreasing sequences always exist in $(\overline{\mathcal{M}}, \leq)$ as recalled in Section 3.3.2. We call the element X of $\overline{\mathcal{M}}$ thus defined the *generalized synchronization trace* of Y.

Adapting the algorithmic point of view to an infinite input introduces obviously some issues, which can be addressed by observing that the output of the Synchronization Algorithm is nondecreasing with its input. We will idealize the situation where the Synchronization Algorithm is repeatedly fed with a nondecreasing input by saying

Algorithm 1 Determines a minimal piece of the synchronizing trace X of $Y = (Y_1, \ldots, Y_N)$

Require: $\overline{Y}_1, \ldots, \overline{Y}_N$ ▷ Recall that $\overline{Y}_i = Y_i \cdot \dagger_i$

for all $i \in \{1, \ldots, N\}$ **do**
  $u_i \leftarrow$ first letter of $\overline{Y}_i$ ▷ u_i is either a real letter or $\dagger_i$
end for
$H \leftarrow \{i \,:\, u_i \neq \text{EOF}\}$ ▷ The set of indices of interest
$\overline{H} \leftarrow \{i \,:\, u_i = \text{EOF}\}$ ▷ The complementary set of H
if $H = \emptyset$ **then** ▷ Case 2a of the above discussion
  return EOF
end if
$K \leftarrow \{i \in H \,:\, u_i \neq \text{WFI}\}$ ▷ The set of indices with real letters
if $K = \emptyset$ **then** ▷ One instance of Case 2(b)i
  return WFI
end if
for all $i \in K$ **do**
  if $\mathcal{R}(u_i) \cap \overline{H} \neq \emptyset$ **then** ▷ No chance to obtain later the expected synchronization for u_i
    $M_i \leftarrow \text{DL}$
  else
    if $u_j = u_i$ for all $j \in \mathcal{R}(u_i)$ **then** ▷ Case where u_i is minimal in all the expected components of Y
      $M_i \leftarrow u_i$
    else
      if $u_j \in \{u_i, \text{WFI}\}$ for all $j \in \mathcal{R}(u_i)$ **then** ▷ Synchronization for u_i is possible in the future
        $M_i \leftarrow \text{WFI}$
      else
        $M_i \leftarrow \text{DL}$ ▷ The piece u_i is not minimal in X
      end if
    end if
  end if
end for
if $M_i = \text{DL}$ for all $i \in K$ **then** ▷ Case 2(b)ii: the synchronization trace X is empty since it has no minimal piece
  return DL
else
  if $M_i = \text{WFI}$ for all $i \in K$ **then** ▷ Case 2(b)i (again)
    return WFI
  else ▷ Case 1
    return one M_i with $M_i \notin \{\text{DL}, \text{WFI}\}$ ▷ Any $M_i \notin \{\text{DL}, \text{WFI}\}$ is minimal in the synchronization trace X
  end if
end if

Algorithm 2 Synchronization Algorithm: computes the synchronization trace of Y

Require: $\overline{Y} = (\overline{Y}_1, \ldots, \overline{Y}_N)$ ▷ $\overline{Y}_i = Y_i \cdot \dagger_i$

1: $X \leftarrow \mathbf{e}$ ▷ Initialize the variable X with the empty heap
2: **call** Algorithm 1 with input $\overline{Y}$
3: $M \leftarrow$ output of Algorithm 1
4: **while** $M \notin \{\text{DL}, \text{WFI}, \text{EOF}\}$ **do**
5: $\quad X \leftarrow X \cdot M$
6: $\quad \overline{Y} \leftarrow M \backslash \overline{Y}$
7: $\quad$ **call** Algorithm 1 with input $\overline{Y}$
8: $\quad M \leftarrow$ output of Algorithm 1
9: **end while**
10: **return** (X, M)

that it is fed with a vector with possibly infinite components. The output, possibly infinite, is defined as the least upper bound in $\overline{\mathcal{M}}$ of the nondecreasing sequence of finite output heaps. It coincides of course with the generalized synchronization trace X defined in (4).

To describe more precisely this algorithmic procedure, consider a vector Y of sequences, some of which may be infinite. We assume that Y is effectively given through some *sampling*, *i.e.*, as an *infinite* concatenation of *finite* vectors $Z_k \in \mathcal{H}$:

$$Y = Z_1 \cdot Z_2 \cdot Z_3 \cdots$$

If a component of Y is finite, the corresponding component of the vector Z_k will be the empty word for k large enough.

Furthermore, we assume that a primitive is able to produce, for each integer $k \geq 1$, a vector $\overline{Z}_k = Z_k \cdot \dagger_k$, where $\dagger_k$ is itself a vector $\dagger_k = (\dagger_{k,i})_i$ with $\dagger_{k,i} \in \{\text{EOF}, \text{WFI}\}$, in such a way that an occurrence of EOF marks the finiteness of the corresponding component of Y. Formally, we assume that the following two properties hold, for all components:

$$\begin{aligned} &(Y_i \text{ is a finite sequence}) \iff (\exists k \geq 1 \quad \dagger_{k,i} = \text{EOF}) \\ \text{and} \quad &\forall k \geq 1 \quad \dagger_{k,i} = \text{EOF} \implies (\forall k' \geq k \quad \dagger_{k',i} = \text{EOF}) \end{aligned}$$

The Generalized Synchronization Algorithm, the pseudo-code of which is given below in Algorithm 3, is then recursively fed with $\overline{Z}_1, \overline{Z}_2, \ldots$, writing out to its output register X. The output of Algorithm 3 is the least upper bound, in $\overline{\mathcal{M}}$, of the sequence of heaps that recursively appear in the register X.

The Generalized Synchronization Algorithm exits its **while** loop if and only if the generalized synchronization trace is finite. In all cases, its output (as defined above) is the generalized synchronization trace of the vector Y, regardless of the decomposition $(\overline{Z}_k)_{k \geq 1}$ that feeds its input.

4—Preliminaries on probabilistic trace monoids

We denote by $\mathbb{R}_+^*$ the set of positive reals.

4.1 — Bernoulli and finite Bernoulli sequences

In this section we collect classical material found in many textbooks [5]. We pay a special attention to presenting this material so as to prepare for its generalization to trace monoids.

Algorithm 3 Generalized Synchronization Algorithm

Require: $(\overline{Z}_k)_{k\geq 1}$ ▷ The Z_ks recursively feed the input
$X \leftarrow \mathbf{e}$ ▷ Initialize the variable X with the empty heap
$k \leftarrow 1$
call Algorithm 2 with input $\overline{Z}_1$
$(U, \dagger) \leftarrow$ output of Algorithm 2 ▷ $\dagger \in \{\text{EOF}, \text{DL}, \text{WFI}\}$
while $\dagger = \text{WFI}$ **do**
  $X \leftarrow X \cdot U$
  $k \leftarrow k + 1$
  call Algorithm 2 with input $\overline{Z}_k$
  $(U, \dagger) \leftarrow$ output of Algorithm 2
end while

4.1.1 *Bernoulli sequences* — Classically, a *Bernoulli sequence* on an alphabet Σ is an infinite sequence $(X_n)_{n\geq 1}$ of independent and identically distributed (*i.i.d.*) random variables, where each X_i takes its values in Σ. In order to eliminate degenerated cases, we assume that the common probability distribution, say ρ, over Σ of all X_i is *positive* on Σ; hence ρ is bound to satisfy:

$$\forall a \in \Sigma \quad \rho_a > 0\,, \qquad\qquad \sum_{a\in\Sigma} \rho_a = 1\,.$$

4.1.2 *Bernoulli measures* — The canonical probability space associated with the Bernoulli sequence $(X_n)_{n\geq 1}$ is the triple $(\partial\Sigma^*, \mathfrak{F}, \mathbb{P})$ defined as follows: the set $\partial\Sigma^*$ is the set of infinite sequences with values in Σ. The σ-algebra $\mathfrak{F}$ is the σ-algebra generated by the countable collection of *elementary cylinders* $\uparrow x$, for x ranging over the free monoid Σ^*, and defined by

$$\uparrow x = \{\omega \in \partial\Sigma^* \ :\ x \leq \omega\}\,,$$

where $x \leq \omega$ means that the infinite sequence ω starts with the finite word x. Finally, $\mathbb{P}$ is the unique probability measure on $(\partial\Sigma^*, \mathfrak{F})$ which takes the following values on elementary cylinders:

$$\forall x \in \Sigma^* \quad \mathbb{P}(\uparrow x) = f(x)\,;$$

here $f : \Sigma^* \to \mathbb{R}_+^*$ is the unique positive function satisfying:

$$\forall a \in \Sigma \quad f(a) = \rho_a\,, \qquad\qquad \forall x, y \in \Sigma^* \quad f(xy) = f(x)f(y)\,, \tag{5}$$

where xy denotes the concatenation of words x and y. Bernoulli sequences correspond exactly to probability measures $\mathbb{P}$ on $(\partial\Sigma^*, \mathfrak{F})$ with the following property:

$$\forall x \in \Sigma^* \quad \mathbb{P}(\uparrow x) > 0\,,$$
$$\forall x, y \in \Sigma^* \quad \mathbb{P}\big(\uparrow (xy)\big) = \mathbb{P}(\uparrow x)\mathbb{P}(\uparrow y)\,.$$

Such probability measures on $(\partial\Sigma^*, \mathfrak{F})$ are called *Bernoulli measures*.

4.1.3 *Uniform Bernoulli measure* — Among Bernoulli measures associated with the alphabet Σ, one and only one is *uniform*, in the following sense:

$$\forall x, y \in \Sigma^* \quad |x| = |y| \implies \mathbb{P}(\uparrow x) = \mathbb{P}(\uparrow y)\,,$$

where $|x|$ denotes the length of the word x. It is characterized by $\mathbb{P}(\uparrow x) = p_0^{|x|}$, where $p_0 = 1/|\Sigma|$.

4.1.4 *Finite Bernoulli sequences* — Let $\rho = (\rho_a)_{a\in\Sigma}$ be a sub-probability distribution over Σ, hence bound to satisfy:

$$\forall a \in \Sigma \quad \rho_a > 0\,, \qquad\qquad \sum_{a\in\Sigma} \rho_a < 1\,.$$

To each sub-probability distribution ρ we associate the function $f : \Sigma^* \to \mathbb{R}$ defined as in (5), and also the following quantities:

$$\varepsilon = 1 - \sum_{a\in\Sigma} \rho_a\,, \qquad\qquad Z = \sum_{x\in\Sigma^*} f(x) = \frac{1}{\varepsilon} < \infty\,.$$

Finally we define the *sub-Bernoulli measure* ν_ρ as the probability distribution over the countable set Σ^*, equipped with the discrete σ-algebra, and defined by:

$$\forall x \in \Sigma^* \quad \nu_\rho(\{x\}) = \frac{1}{Z} f(x)\,.$$

A *finite Bernoulli sequence* is the random sequence of letters that compose a word $x \in \Sigma^*$, drawn at random according to the probability measure ν_ρ on Σ^*.

An *effective* way to produce a finite Bernoulli sequence according to a sub-probability distribution $p = (p_i)_i$ is the following. Consider a stopping symbol EOF, and extend p to a probability distribution by setting $p(\text{EOF}) = 1 - \sum_i p_i$. Then output a Bernoulli sequence according to p, until the symbol EOF first occurs. The letters before the first occurrence of EOF form the sought finite sequence.

4.1.5 *Full elementary cylinders* — It is convenient to consider the following completion of Σ^*:

$$\overline{\Sigma^*} = \Sigma^* \cup \partial\Sigma^*\,.$$

Hence both Bernoulli measures and sub-Bernoulli measures are now defined on the same space $\overline{\Sigma^*}$. For each word $x \in \Sigma^*$, we define the *full elementary cylinder* $\Uparrow x$ as follows:

$$\Uparrow x = \{\xi \in \overline{\Sigma^*} \ :\ x \le \xi\}\,.$$

Here, $x \le \xi$ has the same meaning as above if ξ is an infinite sequence; and it means that x is a prefix of ξ if ξ is a finite word. We gather the description of both Bernoulli and sub-Bernoulli measures in the following result.

• Theorem 4.1—*Let $\mathbb{P}$ be a probability measure on $\overline{\Sigma^*} = \Sigma^* \cup \partial\Sigma^*$. Assume that the function $f : \Sigma^* \to \mathbb{R}$ defined by $f(x) = \mathbb{P}(\Uparrow x)$ is positive multiplicative, that is to say, satisfies:*

$$\forall x \in \Sigma^* \quad f(x) > 0$$
$$\forall x, y \in \Sigma^* \quad f(xy) = f(x)f(y)\,.$$

Define $\rho = (\rho_a)_{a\in\Sigma}$ and ε by:

$$\forall a \in \Sigma \quad \rho_a = f(a)\,, \qquad\qquad \varepsilon = 1 - \sum_{a\in\Sigma} \rho_a\,.$$

Then one and only one of the two following possibilities occurs:

1. *$\varepsilon = 0$. In this case, $\mathbb{P}$ is concentrated on $\partial\Sigma^*$, and characterized by:*

$$\forall x \in \Sigma^* \quad \mathbb{P}(\uparrow x) = \mathbb{P}(\Uparrow x) = f(x)\,.$$

The series $\sum_{x\in\Sigma^*} f(x)$ is divergent, and $\mathbb{P}$ is a Bernoulli measure.

2. $\varepsilon > 0$. *In this case,* $\mathbb{P}$ *is concentrated on* Σ^*. *The series* $Z = \sum_{x\in\Sigma^*} f(x)$ *is convergent, and satisfies:*

$$\varepsilon = \frac{1}{Z}\,, \qquad\qquad \forall x \in \Sigma^* \quad \mathbb{P}(\{x\}) = \varepsilon f(x)\,.$$

The measure $\mathbb{P}$ *is a sub-Bernoulli measure.*

4.2 — Bernoulli and sub-Bernoulli measures on trace monoids

The notions of Bernoulli measure and of sub-Bernoulli measure extend to trace monoids the notions of Bernoulli sequences and of finite Bernoulli sequences. They provide a theoretical ground for concurrency probabilistic models, in the framework of trace monoids.

4.2.1 *Valuations* — Let $\mathbb{R}_+^*$ be equipped with the monoid structure $(\mathbb{R}_+^*, \times, 1)$. A *valuation* on a trace monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$ is a morphism of monoids $f : \mathcal{M} \to \mathbb{R}_+^*$. It is thus a function $f : \mathcal{M} \to \mathbb{R}_+^*$ satisfying:

$$f(\mathbf{e}) = 1\,, \qquad\qquad \forall x, y \in \mathcal{M} \quad f(x \cdot y) = f(x)f(y).$$

4.2.2 *Möbius transform; Möbius and sub-Möbius valuations* — Let $f : \mathcal{M} \to \mathbb{R}$ be a valuation. The *Möbius transform* of f is the function $h : \mathscr{C} \to \mathbb{R}$ defined by:

$$\forall c \in \mathscr{C} \quad h(c) = \sum_{c' \in \mathscr{C}\,:\,c' \geq c} (-1)^{|c'|-|c|} f(c'), \quad \text{only defined on cliques.} \tag{6}$$

An alternative expression for the Möbius transform is the following. For each clique $c \in \mathscr{C}$, let $\mathcal{M}^{(c)}$ be the sub-trace monoid generated by those letters $a \in \Sigma$ such that $a \parallel c$. Here, $a \parallel c$ reads as "a parallel to c", and means that $(a, b) \in I$ for all letters b that occur in the clique c. Then:

$$h(c) = f(c) \cdot \mu_{\mathcal{M}^{(c)}}(t_1, \ldots, t_N)\,, \qquad\qquad \text{with } t_i = f(a_i)\,, \tag{7}$$

where $\mu_{\mathcal{M}^{(c)}}$ denotes the multivariate Möbius polynomial (see Section 3.1.4) of the trace monoid $\mathcal{M}^{(c)}$. By convention, the expression $\mu_{\mathcal{M}^{(c)}}(t_1, \ldots, t_N)$ actually involves only the variables t_i associated with those generators belonging to $\mathcal{M}^{(c)}$.

In particular, $h(\mathbf{e}) = \mu_{\mathcal{M}}(t_1, \ldots, t_N)$—that was already observable on (6).

A valuation f is said to be [2]:

1. A *Möbius* valuation if:

$$h(\mathbf{e}) = 0\,, \qquad\qquad \forall c \in \mathfrak{C} \quad h(c) > 0\,. \tag{8}$$

2. A *sub-Möbius* valuation if:

$$h(\mathbf{e}) > 0\,, \qquad\qquad \forall c \in \mathfrak{C} \quad h(c) > 0\,. \tag{9}$$

Equivalently, as seen from (7), if $\Sigma = \{a_1, \ldots, a_N\}$ and if we put $t_i = f(a_i)$ for $i \in \{1, \ldots, N\}$, then f is:

1. A *Möbius* valuation if:

$$\mu_{\mathcal{M}}(t_1, \ldots, t_N) = 0\,, \qquad\qquad \forall c \in \mathfrak{C} \quad \mu_{\mathcal{M}^{(c)}}(t_1, \ldots, t_N) > 0\,. \tag{10}$$

2. A *sub-Möbius* valuation if:

$$\mu_{\mathcal{M}}(t_1, \ldots, t_N) > 0\,, \qquad\qquad \forall c \in \mathfrak{C} \quad \mu_{\mathcal{M}^{(c)}}(t_1, \ldots, t_N) > 0\,. \tag{11}$$

4.2.3 *Cylinders and σ-algebras on $\overline{\mathcal{M}}$ and on $\partial\mathcal{M}$* — Recall that we have introduced infinite traces in Section 3.3, yielding the completion $\overline{\mathcal{M}} = \mathcal{M} \cup \partial\mathcal{M}$.

To each trace $x \in \mathcal{M}$, we associate the *elementary cylinder* $\uparrow x \subseteq \partial\mathcal{M}$ and the *full elementary cylinder* $\Uparrow x \subseteq \overline{\mathcal{M}}$, defined as follows:

$$\Uparrow x = \{\xi \in \overline{\mathcal{M}} \ : \ x \leq \xi\}\,, \qquad \uparrow x = \{\xi \in \partial\mathcal{M} \ : \ x \leq \xi\} = \Uparrow x \cap \partial\mathcal{M}\,.$$

The set $\overline{\mathcal{M}}$ is equipped with the σ-algebra $\overline{\mathfrak{F}}$ generated by the collection of full elementary cylinders, and the set $\partial\mathcal{M}$ is equipped with the σ-algebra $\mathfrak{F}$ induced by $\overline{\mathfrak{F}}$ on $\partial\mathcal{M}$. Both σ-algebras are Borel σ-algebras for compact and metrisable topologies. The restriction of $\overline{\mathfrak{F}}$ to $\mathcal{M}$ is the discrete σ-algebra.

4.2.4 *Multiplicative probability measures* — In order to state an analogous result to Theorem 4.1 for trace monoids, we introduce the following definition, borrowed from [2].

• Definition 4.2—*A probability measure $\mathbb{P}$ on $(\overline{\mathcal{M}}, \overline{\mathfrak{F}})$ is said to be* multiplicative *whenever it satisfies the following property:*

$$\forall x \in \mathcal{M} \quad \mathbb{P}(\Uparrow x) > 0\,,$$
$$\forall x, y \in \mathcal{M} \quad \mathbb{P}\big(\Uparrow (x \cdot y)\big) = \mathbb{P}(\Uparrow x) \cdot \mathbb{P}(\Uparrow y)\,.$$

We define the valuation $f : \mathcal{M} \to \mathbb{R}$ associated with $\mathbb{P}$ and the number ε by:

$$\forall x \in \mathcal{M} \quad f(x) = \mathbb{P}(\Uparrow x)\,, \qquad\qquad \varepsilon = h(\mathbf{e})\,,$$

where $h : \mathscr{C} \to \mathbb{R}$ is the Möbius transform of f.

The relationship between multiplicative measures and Möbius and sub-Möbius valuations is as follows [2, 3].

• Theorem 4.3—*There is a bijective correspondence between multiplicative measures $\mathbb{P}$ and valuations which are either Möbius or sub-Möbius. The alternative is the following:*

1. *$\varepsilon = 0$. The valuation is Möbius. In this case, $\mathbb{P}$ is concentrated on $\partial\mathcal{M}$ and characterized by:*

$$\forall x \in \mathcal{M} \quad \mathbb{P}(\uparrow x) = f(x)\,.$$

The series $\sum_{x\in\mathcal{M}} f(x)$ is divergent. We say that $\mathbb{P}$ is a Bernoulli measure.

2. *$\varepsilon > 0$. The valuation is sub-Möbius. In this case, $\mathbb{P}$ is concentrated on $\mathcal{M}$ and satisfies:*

$$\forall x \in \mathcal{M} \quad \mathbb{P}(\{x\}) = \varepsilon f(x)\,.$$

The series $Z = \sum_{x\in\mathcal{M}} f(x)$ is convergent and satisfies $Z = 1/\varepsilon$. We say that $\mathbb{P}$ is a sub-Bernoulli measure.

4.2.5 *Uniform multiplicative measures* — A valuation $f : \mathcal{M} \to \mathbb{R}$ is said to be *uniform* if $f(a)$ is constant, for a ranging over Σ. This is equivalent to saying that $f(x)$ only depends on the length of x, and also equivalent to saying that $f(x) = p^{|x|}$ for some real p.

A multiplicative measure is *uniform* if the associated valuation is uniform. The following result describes uniform multiplicative measures [2, 3]. They are related to the root of smallest modulus of the Möbius polynomial of the trace monoid (see Section 3.1.3).

• Theorem 4.4—*Uniform multiplicative measures $\mathbb{P}$ on a trace monoid $\mathcal{M}$ are in bijection with the half closed interval $(0, p_0]$, where p_0 is the root of smallest modulus of $\mu_{\mathcal{M}}$. The correspondence associates with $\mathbb{P}$ the unique real p such that $\mathbb{P}(\Uparrow x) = p^{|x|}$ for all $x \in \mathcal{M}$.*

The alternative is the following:

1. *$p = p_0$. In this case, $\mathbb{P}$ is Bernoulli (concentrated on the boundary).*
2. *$p < p_0$. In this case, $\mathbb{P}$ is sub-Bernoulli (concentrated on the monoid).*

4.2.6 *Extension and restriction of valuations* — We shall need the following result for the construction of the PFSA in Section 6.3.

• Theorem 4.5—*Let $\mathcal{M} = \mathcal{M}(\Sigma, I)$ be an irreducible trace monoid.*

1. *Let $f : \mathcal{M} \to \mathbb{R}_+^*$ be a Möbius valuation, let Σ' be any proper subset of Σ and let $\mathcal{M}'$ be the submonoid of $\mathcal{M}$ generated by Σ'. Then the restriction $f' : \mathcal{M}' \to \mathbb{R}_+^*$ of f to $\mathcal{M}'$ is a sub-Möbius valuation.*
2. *Let $\Sigma' = \Sigma \setminus \{a\}$, where a is any element of Σ, let $\mathcal{M}'$ be the submonoid of $\mathcal{M}$ generated by Σ', and let $f' : \mathcal{M}' \to \mathbb{R}_+^*$ be a sub-Möbius valuation. Then there exists a unique Möbius valuation $f : \mathcal{M} \to \mathbb{R}_+^*$ that extends f' on $\mathcal{M}'$.*

Proof. Proof of point 1. Let f, f' and Σ' be as in the statement. Let also h and h' denote the Möbius transforms of f and of f'. Let S be the series with nonnegative terms:

$$S = \sum_{x \in \mathcal{M}'} f(x).$$

We claim that this series is convergent. To prove it, recall from [6] that a pair (γ, γ') of cliques is said to be in normal form, denoted by $\gamma \to \gamma'$, if $\forall b \in \gamma' \; \exists a \in \gamma \; (a, b) \notin I$. Any trace $x \in \mathcal{M}$ can be uniquely written as a product $x = \gamma_1 \cdot \ldots \cdot \gamma_k$ of cliques such that $\gamma_i \to \gamma_{i+1}$ holds for all $i = 1, \ldots, k-1$.

Let the nonnegative matrices $A = (A_{\gamma,\gamma'})_{(\gamma,\gamma') \in \mathfrak{C} \times \mathfrak{C}}$ and $B = (B_{\gamma,\gamma'})_{(\gamma,\gamma') \in \mathfrak{C} \times \mathfrak{C}}$, where $\mathfrak{C}$ denotes the set of nonempty cliques of $\mathcal{M}$, be defined by:

$$A_{\gamma,\gamma'} = \mathbf{1}(\gamma \in \mathcal{M}') \cdot \mathbf{1}(\gamma' \in \mathcal{M}') \cdot \mathbf{1}(\gamma \to \gamma') \cdot f(\gamma'), \quad B_{\gamma,\gamma'} = \mathbf{1}(\gamma \to \gamma') \cdot f(\gamma').$$

It follows from the existence and uniqueness of the normal form for traces, decomposing the traces $x \in \mathcal{M}$ according to the number of terms of their normal form, that the series S writes as:

$$S = 1 + \sum_{k \geq 1} I \cdot A^k \cdot J = 1 + I \cdot \Big(\sum_{k \geq 1} A^k \Big) \cdot J,$$

where I and J are row and column vectors filled with 1s. We know by [1, Lemma 6.4] that the matrix B has spectral radius 1, since f is assumed to be Möbius, and that it is a primitive matrix since $\mathcal{M}$ is irreducible. But $A \leq B$ with $A \neq B$. Therefore, it follows from the Perron-Frobenius Theorem [21] that A has spectral radius < 1 and thus that the series S is convergent.

By the Möbius inversion formula [22], it entails that the relation $\big(\sum_{x \in \mathcal{M}'} f(x)\big) \cdot h'(\mathbf{e}) = 1$ holds in the fields of reals. Hence $h'(\mathbf{e}) > 0$. It remains to prove that $h'(\delta) > 0$ also holds for any non empty clique δ of $\mathcal{M}'$. This is a bit easier to prove. For any non empty clique δ of $\mathcal{M}'$, let $\mathcal{M}'^{(\delta)}$ and $\mathcal{M}^{(\delta)}$ be the submonoids of $\mathcal{M}'$

and of $\mathcal{M}$ respectively, be defined as in Section 4.2.2. Then $\mathcal{M}'^{(\delta)} \subseteq \mathcal{M}^{(\delta)}$, hence the following inequalities between series with nonnegative terms hold:

$$\sum_{x \in \mathcal{M}'^{(\delta)}} f(x) \leq \sum_{x \in \mathcal{M}^{(\delta)}} f(x) = \frac{1}{h(\delta)} < \infty.$$

Therefore, as above, the equality $h'(\delta) \cdot \left(\sum_{x \in \mathcal{M}'^{(\delta)}} f(x)\right) = 1$ holds in the field of reals, proving that $h'(\delta) > 0$, which completes the proof that f' is a sub-Möbius valuation.

Proof of point 2. Let a, Σ' and $f' : \mathcal{M}' \to \mathbb{R}_+^*$ be as in the statement. Let also $h' : \mathscr{C}' \to \mathbb{R}$ the Möbius transform of f', where $\mathscr{C}'$ is the set of cliques of $\mathcal{M}'$. Assuming that a Möbius extension $f : \mathcal{M} \to \mathbb{R}_+^*$ of f' exists, we first prove its uniqueness. For each positive real t, let $f_t : \mathcal{M} \to \mathbb{R}$ be the valuation defined by $f_t(a) = t$ and $f_t(\alpha) = f'(\alpha)$ for $\alpha \in \Sigma'$. Then any valuation on $\mathcal{M}$ extending f' is of the form f_t for some t. Let $h_t : \mathscr{C} \to \mathbb{R}$ denote the Möbius transform of f_t. We evaluate $h_t(\mathbf{e})$ as follows:

$$h_t(\mathbf{e}) = \sum_{\gamma \in \mathscr{C} \,:\, a \in \gamma} (-1)^{|\gamma|} f_t(\gamma) + \sum_{\gamma \in \mathscr{C} \,:\, a \notin \gamma} (-1)^{|\gamma|} f_t(\gamma).$$

On the one hand, observing the equality of sets $\mathscr{C}' = \{\gamma \in \mathscr{C} \;:\; a \notin \gamma\}$, we recognize $h'(\mathbf{e})$ in the second sum. On the other hand, using the notation already introduced $a \parallel \gamma$, for $\gamma \in \mathscr{C}$, to denote that $a \notin \gamma$ and $a \cdot \gamma \in \mathscr{C}$, the range of the first sum above is in bijection with the set of cliques $\delta \in \mathscr{C}'$ such that $a \parallel \delta$, the bijection associating δ with $\gamma = a \cdot \delta$. We then have $(-1)^{|\gamma|} f_t(\gamma) = (-t) \cdot (-1)^{|\delta|} f'(\delta)$. Henceforth:

$$h_t(\mathbf{e}) = (-t) \cdot K + h'(\mathbf{e}), \quad \text{with } K = 1 - K' \text{ and } K' = \sum_{\delta \in \mathscr{C}' \,:\, a \| \delta \wedge \delta \neq \mathbf{e}} (-1)^{|\delta|+1} f'(\delta).$$

Let X be a random trace associated with the sub-Möbius valuation f'. We evaluate the probability of the event $U = \{\exists b \in \Sigma' \;:\; a \parallel b \wedge a \leq X\}$. The inclusion-exclusion principle yields:

$$\mathbb{P}(U) = \sum_{\delta \in \mathscr{C}' \,:\, a \| \delta \wedge \delta \neq \mathbf{e}} (-1)^{|\delta|+1} \mathbb{P}(X \geq \delta) = K',$$

the later equality since $\mathbb{P}(X \geq \delta) = f'(\delta)$ by definition of X. The event U has probability less than 1, otherwise all pieces of Σ' would be parallel to a, contradicting that $\mathcal{M}$ is irreducible. We deduce that $K' \in [0, 1)$ and thus $K \in (0, 1]$. In particular, if f_t is Möbius, it entails that $h_t(\mathbf{e}) = 0$, and since we have just seen that $K \neq 0$, it implies that $t = h'(\mathbf{e})/K$, proving the sought uniqueness.

Let us now prove the existence of a Möbius extension f. Let $\mathcal{M}^{(\delta)}$ denote as above, for any clique $\delta \in \mathscr{C}$, the sub-monoid of $\mathcal{M}$ generated by those letters $b \in \Sigma$ such that $b \parallel \delta$. For each $\delta \in \mathscr{C}$, we introduce the formal series:

$$G_\delta(t) = \sum_{x \in \mathcal{M}^{(\delta)}} f_t(x).$$

Now, let t_0 be the radius of convergence of the power series $G_{\mathbf{e}}(t) = \sum_{x \in \mathcal{M}} f_t(x)$. Since all the power series $G_\delta(t)$ have non negative coefficients, they satisfy $G_\delta(t) \leq G_{\mathbf{e}}(t) < \infty$ for all $t \in [0, t_0)$. In particular, the radius of convergence r_δ of G_δ satisfies $r_\delta \geq t_0$. Actually, reasoning as in the proof of point 1 and invoking the Perron-Frobenius Theorem, we see that the strict inequality $r_\delta > t_0$ holds for all $\delta \neq \mathbf{e}$. Therefore, for all $\delta \neq \mathbf{e}$, the series $G_\delta(t_0)$ is convergent, and thus the Móbius inversion formula yields the following equality in the field of reals: $G_\delta(t_0) \cdot h_{t_0}(\delta) = 1$. We conclude that $h_{t_0}(\delta) > 0$ for all cliques $\delta \neq \mathbf{e}$. Since $h_{t_0}(\mathbf{e}) = 0$, we conclude that f_{t_0} is the sought Möbius valuation extending f'. □

5—The Probabilistic Synchronization Algorithm

In this section we consider a network of alphabets sharing some common letters. We then wish to generate random traces of the synchronization monoid, in a distributed and incremental way. The first idea that comes in mind is to generate local Bernoulli sequences, and to see what is the synchronization trace of these sequences. This constitutes the Probabilistic Synchronization Algorithm, which is thus a probabilistic variant of the Generalized Synchronization Algorithm which was described in Algorithm 3.

The random traces thus obtained can be either finite or infinite. In all cases their probability distribution is multiplicative, hence the theory of Bernoulli and sub-Bernoulli measures for trace monoids is the adequate tool for their study. After having established this rather easy result, we turn to specific examples. We obtain the non trivial result that for path models, any Bernoulli or sub-Bernoulli measure can be generated by this simple technique.

5.1 — Description of the PSA

Let $(\Sigma_1, \ldots, \Sigma_N)$ be a network of N alphabets with $N \geq 2$, let $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_N$, and let $\mathcal{M} = \mathcal{M}(\Sigma, I)$ be the synchronization trace monoid, as described in Section 3.4.1. Recall that we identify $\mathcal{M}$ with the sub-monoid $\mathscr{G} \subseteq \mathcal{H}$ defined in Section 3.4.1, and that indices in $1, \ldots, N$ are seen as resources.

Assume that, to each resource $i \in \{1, \ldots, N\}$, is attached a device able to produce a Σ_i-Bernoulli sequences Y_i, either finite or infinite, with a specified probability or sub-probability distribution p_i on Σ_i . As we have seen in Section 4.1.4, the generation of such Bernoulli sequences is effective, together with the information that the sequence is over in case where p_i is a sub-probability distribution.

Henceforth, it is straightforward for each device to produce a sampling of Y_i under the form $Z_{i,1} \cdot Z_{i,2} \cdot \ldots$, and moreover to tag each sub-sequence $Z_{i,k}$ with an additional symbol $\dagger_{i,k} \in \{\text{EOF}, \text{WFI}\}$ in order to deliver a sequence $(\overline{Z}_{i,k})_{k \geq 1}$ with $\overline{Z}_{i,k} = Z_{i,k} \cdot \dagger_{i,k}$ as specified in Section 3.4.6. The symbol $\dagger_{i,k}$ is given the value WFI until the device decides the sequence is over (if it ever does), after which the symbol $\dagger_{i,k}$ is given the value EOF.

The Probabilistic Synchronization Algorithm (PSA), the pseudo-code of which is given below in Algorithm 4, consists in executing in parallel both the local generation of the sequences $Y_1, \ldots, Y_N$, and the Generalized Synchronization Algorithm described previously in Algorithm 3.

Algorithm 4 Probabilistic Synchronization Algorithm

Require: $p_1, \ldots, p_N$ ▷ Probability or sub-probability distributions

```
while Algorithm 3 does not exit do
    for all i ∈ {1, ..., N} do
        for all k = 1, 2, ... do
            generate the k^th sampling Z̄_{i,k} of a Bernoulli sequence according to p_i
            feed Algorithm 3 with Z̄_{i,k}
        end for
    end for
end while
```

5.2 — Analysis of the algorithm

5.2.1 *Distribution of* PSA *random traces* — The trace $y \in \overline{\mathcal{M}}$, output of the execution of the PSA (Algorithm 4), is random. What is its distribution?

• Theorem 5.1—*The probability distribution* $\mathbb{P}$ *of the random trace produced by the* PSA *is a multiplicative probability measure on* $\overline{\mathcal{M}}$. *The valuation* $f : \mathcal{M} \to \mathbb{R}$ *associated with this measure by* $f(x) = \mathbb{P}(\Uparrow x)$ *is such that:*

$$\forall a \in \Sigma \quad f(a) = \prod_{i \in R(a)} p_i(a)\,, \tag{12}$$

where p_i *is the probability or sub-probability distribution on* Σ_i *, and* $R(a)$ *is the set of resources associated with* a.

Proof. For each letter $a \in \Sigma$, let q_a be the real number defined by:

$$q_a = \prod_{i \in R(a)} p_i(a)\,.$$

Let $y \in \overline{\mathcal{M}}$ be the random trace produced by the PSA. Fix $z \in \mathcal{M}$ a trace, and let $(z_1, \ldots, z_N)$ be the representation of z in $\mathscr{G}$. Then $y \geq z$ holds if and only if, for every index $i \in \{1, \ldots, N\}$, the corresponding sequence x_i starts with z_i .

Let $z = a_1 \cdot \ldots \cdot a_j$ be any decomposition of z as a product of generators $a_k \in \Sigma$. Since each sequence x_i produced by the device number i is Bernoulli or sub-Bernoulli with distribution or sub-distribution p_i , and since the sequences are mutually independent, we have thus:

$$\mathbb{P}(y \geq z) = \prod_{k=1}^{j} q_{a_k}\,.$$

In other words, if $f : \mathcal{M} \to \mathbb{R}$ is the valuation defined by (12), one has $\mathbb{P}(\Uparrow z) = f(z)$. This shows that $\mathbb{P}$ is multiplicative. □

5.2.2 *Small values* — We shall see that not all multiplicative probabilities on $\overline{\mathcal{M}}$ can be reached by this technique. However, as long as only "small values" are concerned, the PSA can reach any target multiplicative measure. Let us first introduce the following convention: if $f : \mathcal{M} \to \mathbb{R}$ is a valuation, and if ε is a real number, we denote by εf the valuation which values on generators are given by:

$$\forall a \in \Sigma \quad (\varepsilon f)(a) = \varepsilon f(a)\,.$$

Hence the value of εf on arbitrary traces are given by:

$$\forall x \in \mathcal{M} \quad (\varepsilon f)(x) = \varepsilon^{|x|} f(x)\,.$$

The following result states that, at the expense of having small synchronizing traces, the relative frequency of letters in traces produced by the PSA suffers from no constraint.

• Proposition 5.2—*Let* $(\Sigma_1, \ldots, \Sigma_N)$ *be* N *alphabets, let* $\mathcal{M} = \mathcal{M}(\Sigma, I)$ *be the synchronization trace monoid, and let* $t : \mathcal{M} \to \mathbb{R}$ *be a valuation on* $\mathcal{M}$. *Then there exists* $\varepsilon > 0$ *and sub-probability distributions* $(p_i)_{1 \leq i \leq N}$ *, with* p_i *a sub-probability distribution on* Σ_i *, such that the valuation* f *characterizing the random trace* $y \in \mathcal{M}$ *produced by the* PSA *with respect to* $(p_i)_{1 \leq i \leq N}$ *, satisfies:*

$$f = \varepsilon t\,.$$

Proof. For $\varepsilon > 0$ to be specified later, let $(p_i)_{1 \le i \le N}$ be the family of real valued functions, $p_i : \Sigma_i \to \mathbb{R}$, defined by:

$$\forall a \in \Sigma_i \quad p_i(a) = \big(\varepsilon t(a)\big)^{1/|R(a)|} .$$

For $\varepsilon > 0$ small enough, all p_i are sub-probability distributions over Σ_i. According to Th. 5.1, the valuation f describing the distribution of the random trace produced by the PSA satisfies:

$$f(a) = \prod_{i \in R(a)} p_i(a) = \varepsilon t(a) .$$

The proof is complete. □

5.2.3 *Reduction to irreducible trace monoids* — Assume that the trace monoid is not irreducible. Hence the dependence relation (Σ, D) has several connected components, let us say that it has two components (S_1, D_1) and (S_2, D_2) to simplify the discussion.

Let I_1 and I_2 be the dependence relations on S_1 and S_2 defined by:

$$I_1 = (S_1 \times S_1) \cap I = (S_1 \times S_1) \setminus D_1 , \quad I_2 = (S_2 \times S_2) \cap I = (S_2 \times S_2) \setminus D_2 .$$

Putting $\mathcal{M}_1 = \mathcal{M}(S_1, I_1)$ and $\mathcal{M}_2 = \mathcal{M}(S_2, I_2)$, we have:

$$\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 , \qquad \overline{\mathcal{M}} = \overline{\mathcal{M}_1} \times \overline{\mathcal{M}_2} .$$

It follows from Theorem 5.1 that the distribution $\mathbb{P}$ of the random trace y produced by the PSA is a tensor product $\mathbb{P}_1 \otimes \mathbb{P}_2$; probabilistically speaking, y is obtained as the concatenation of two independent traces $y_1 \in \mathcal{M}_1$ and $y_2 \in \mathcal{M}_2$. The probability measures $\mathbb{P}_1$ and $\mathbb{P}_2$, of y_1 and y_2 respectively, are identical as those deriving from PSA algorithms restricted to the alphabets concerning S_1 and S_2 respectively.

In conclusion, the PSA algorithm decomposes as a product of sub-PSA algorithms on the irreducible components of the synchronization trace monoid. Hence there is no loss of generality in assuming, for the sake of analysis, that the synchronization trace monoid is irreducible.

5.3 — Example: the path model

The path model is close to the dimer model, a topic of numerous studies in Combinatorics and in Statistical Physics [23, 15]. Here we shall see that the path model is a framework where the PSA works at its best, in the sense that any multiplicative measure on $\overline{\mathcal{M}}$ can be obtained through the PSA.

The path model is defined as the trace monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$ on $N+1$ generators: $\Sigma = \{a_0, \ldots, a_N\}$, where $N \ge 0$ is a fixed integer. The independence relation is defined by:

$$I = \{(a_i, a_j) \ : \ |i - j| > 1\} .$$

Then $\mathcal{M}$ is the synchronization monoid of the N-tuple of alphabets $(\Sigma_1, \ldots, \Sigma_N)$ with $\Sigma_i = \{a_{i-1}, a_i\}$ for $i \in \{1, \ldots, N\}$. The set of resources of a_i is:

$$R(a_i) = \begin{cases} \{1\}, & \text{if } i = 0, \\ \{N\}, & \text{if } i = N, \\ \{i, i+1\}, & \text{if } 0 < i < N. \end{cases} \tag{13}$$

Assume that each of the N alphabets $\Sigma_1, \ldots, \Sigma_N$ is equipped with a positive probability distribution, say $p_1, \ldots, p_N$. Then, with probability 1, all tuples $(x_1, \ldots, x_N)$,

where x_i is a Bernoulli infinite sequence distributed according to p_i, are synchronizing. Indeed, each x_i contains infinitely many occurrences of a_{i-1} and of a_i, and this is enough to ensure the synchronization. Therefore, the PSA yields a multiplicative measure entirely supported by the set of infinite traces of $\mathcal{M}$, hence a Bernoulli measure on $\partial\mathcal{M}$.

Actually, the following result shows that *every* Bernoulli measure on the path model can be obtained through the execution of the PSA.

• Theorem 5.3—*Let $\mathbb{P}$ be a Bernoulli measure on $\partial\mathcal{M}$, where $\mathcal{M}$ is the synchronization monoid associated with the path model. For $i \in \{0, \ldots, N\}$, put:*

$$t_i = \mathbb{P}(\uparrow a_i)\,.$$

Then there exists a unique tuple $(p_1, \ldots, p_N)$, with p_i a positive probability distribution over $\Sigma_i = \{a_{i-1}, a_i\}$, such that the random infinite trace $\xi \in \partial\mathcal{M}$ produced by the PSA based on $(p_1, \ldots, p_N)$ has the distribution probability $\mathbb{P}$.

The probability distributions $p_1, \ldots, p_N$ are computed recursively by:

$$p_1(a_0) = t_0 \qquad p_1(a_1) = 1 - t_0 \tag{14}$$

$$i \in \{2, \ldots, N\} \quad p_i(a_{i-1}) = \frac{t_{i-1}}{p_{i-1}(a_{i-1})} \qquad p_i(a_i) = 1 - p_i(a_{i-1})\,. \tag{15}$$

Remark. The relations (14)–(15) are necessary conditions, almost immediate to establish from the result of Theorem 5.1. What is not obvious is that the numbers thus defined stay within $(0, 1)$, yielding indeed probability distributions $\big(p_i(a_{i-1}) \quad p_i(a_i)\big)$ over Σ_i, and that $p_N(a_N) = t_N$.

Proof. Let $f : \mathcal{M} \to \mathbb{R}$ be the valuation associated with $\mathbb{P}$. According to Theorem 4.3, f is a Möbius valuation—we will use this fact in a moment.

Proof of uniqueness of $(p_1, \ldots, p_N)$ and proof of (14)–(15). With respect to a tuple $(p_1, \ldots, p_N)$ of positive probability distributions over $(\Sigma_1, \ldots, \Sigma_N)$, the PSA produces a random infinite trace $\xi \in \partial\mathcal{M}$. Let $g : \mathcal{M} \to \mathbb{R}$ be the valuation associated with the probability distribution of ξ. Then, according to Theorem 5.1, one has:

$$\forall i \in \{0, \ldots, N\} \quad g(a_i) = \prod_{r \in R(a_i)} p_r(a_i)\,.$$

Referring to (13), $f = g$ is equivalent to:

$$t_0 = p_1(a_0) \tag{16}$$

$$t_N = p_N(a_N) \tag{17}$$

$$0 < i < N \qquad t_i = p_i(a_i)p_{i+1}(a_i) \tag{18}$$

It follows at once that the tuple $(p_1, \ldots, p_N)$ inducing $\mathbb{P}$ through the PSA, if it exists, is unique and satisfies necessarily the recurrence relations (14)–(15).

Proof of existence of $(p_1, \ldots, p_N)$. Instead of starting from the recurrence relations (14)–(15), we use a different formulation. For $i \in \{-1, \ldots, N\}$, let $\mathcal{M}_{0,i}$ be the sub-trace monoid of $\mathcal{M}$ generated by $\{a_0, \ldots, a_i\}$. Let also $\mu_{0,i}$ be the evaluation on

$(t_0, \ldots, t_i)$ of the multivariate Möbius polynomial of $\mathcal{M}_{0,i}$ (see Section 3.1.4). Hence:

$$\begin{aligned}
\mu_{0,-1} &= 1 \\
\mu_{0,0} &= 1 - t_0 \\
\mu_{0,1} &= 1 - t_0 - t_1 \\
\mu_{0,2} &= 1 - t_0 - t_1 - t_2 + t_0t_2 \\
\mu_{0,3} &= 1 - t_0 - t_1 - t_2 - t_3 + t_0t_2 + t_0t_3 + t_1t_3 \\
\mu_{0,4} &= 1 - t_0 - t_1 - t_2 - t_3 - t_4 + t_0t_2 + t_0t_3 + t_0t_4 + t_1t_3 + t_1t_4 + t_2t_4 - t_0t_2t_4 \\
&\cdots
\end{aligned}$$

For any $i \in \{-1, \ldots, N-2\}$, the monoid $\mathcal{M}_{0,i}$ coincides with the sub-monoid $\mathcal{M}^{(c_i)}$ as defined in Section 4.2.2, where c_i is the following clique of $\mathcal{M}$:

$$c_i = \begin{cases} a_{i+2} \cdot a_{i+4} \cdot \cdots\cdots \cdot a_N\,, & \text{if } N - i \equiv 0 \mod 2, \\ a_{i+2} \cdot a_{i+4} \cdot \cdots\cdots \cdot a_{N-1}\,, & \text{if } N - i \equiv 1 \mod 2. \end{cases}$$

The clique c_i is non empty as long as $i < N-1$. Therefore, according to (10), the Möbius conditions on f ensure the positivity of all numbers $\mu_{0,i}$ for $i < N-1$.

Let $i \in \{0, \ldots, N-2\}$. Any clique γ of $\mathcal{M}_{0,i+1}$ either belongs to $\mathcal{M}_{0,i}$, or contains an occurrence of a_{i+1}. In the later case, this clique γ is of the form $\gamma = a_{i+1} \cdot \gamma'$, with γ' ranging over cliques of $\mathcal{M}_{0,i-1}$. It follows at once that the following recurrence relation holds:

$$0 \le i < N-1 \quad \mu_{0,i+1} = \mu_{0,i} - t_{i+1}\mu_{0,i-1} \tag{19}$$

Similarly, any clique γ of $\mathcal{M}_{0,N}$ is either contained in $\mathcal{M}_{0,N-2}$, or it contains an occurrence of a_{N-1}, in which case it writes as $\gamma = a_{N-1} \cdot \gamma'$ with γ' ranging over cliques of $\mathcal{M}_{0,N-3}$, or it contains occurrence of a_N, in which case it writes as $\gamma = a_N \cdot \gamma'$ with γ' ranging over cliques of $\mathcal{M}_{0,N-2}$. We deduce: $\mu_{0,N} = \mu_{0,N-2} - t_{N-1}\mu_{0,N-3} - t_N\mu_{0,N-2}$. But $\mu_{0,N}$ is the evaluation on $(t_0, \ldots, t_N)$ of the multivariate Möbius polynomial of $\mathcal{M} = \mathcal{M}_{0,N}$, hence $\mu_{0,N} = 0$ since f is a Möbius valuation. We obtain:

$$(1 - t_N)\mu_{0,N-2} - t_{N-1}\mu_{0,N-3} = 0. \tag{20}$$

Now, since all the numbers $\mu_{0,i}$ for $i \in \{-1, \ldots, N-2\}$ are positive, we define the family $(p_i)_{1 \le i \le N}$ as follows:

$$p_1(a_0) = t_0 \qquad p_1(a_1) = 1 - t_0 \tag{21}$$

$$1 < i < N \quad p_i(a_{i-1}) = t_{i-1}\frac{\mu_{0,i-3}}{\mu_{0,i-2}} \qquad p_i(a_i) = \frac{\mu_{0,i-1}}{\mu_{0,i-2}} \tag{22}$$

$$p_N(a_{N-1}) = 1 - t_N \qquad p_N(a_N) = t_N \tag{23}$$

All numbers appearing in (21), (22) and (23) are positive, and equations (16) and (17) are satisfied. As for (18), we write, for $1 < i < N-1$:

$$p_i(a_i)p_{i+1}(a_i) = \frac{\mu_{0,i-1}}{\mu_{0,i-2}}t_i\frac{\mu_{0,i-2}}{\mu_{0,i-1}} = t_i$$

For $i = N-1$, we have:

$$p_{N-1}(a_{N-1})p_N(a_{N-1}) = \frac{\mu_{0,N-2}}{\mu_{0,N-3}}(1 - t_N) = t_{N-1},$$

the later equality by (20). We have shown so far that (16), (17) and (18) are satisfied.

It remains to see that all $\begin{pmatrix} p_i(a_{i-1}) & p_i(a_i) \end{pmatrix}$, for i ranging over $\{2, \ldots, N-1\}$, are positive probability vectors, since this is trivially true for $i = 1$ and for $i = N$. We have already observed that they are all positive vectors. For $1 < i < N$, one has:

$$p_i(a_i) + p_i(a_{i-1}) = \frac{\mu_{0,i-1} + t_{i-1}\mu_{0,i-3}}{\mu_{0,i-2}} = 1$$

by virtue of (19). Hence each p_i is indeed a positive probability distribution on Σ_i, which completes the proof. □

Theorem 5.3 can be adapted to the case of sub-Bernoulli measures instead of Bernoulli measures, still for the path model, as follows. In a nutshell: every sub-Bernoulli measure can be simulated by synchronization of sub-Bernoulli sequences, but there is no uniqueness in the choice of the local sub-probability distributions.

• Theorem 5.4—*Let $\mathbb{P}$ be a sub-Bernoulli measure on $\mathcal{M}$, where $\mathcal{M}$ is the synchronization monoid associated with the path model. For $i \in \{0, \ldots, N\}$, put:*

$$t_i = \mathbb{P}(\uparrow a_i)\,.$$

Then there exists a tuple $(p_1, \ldots, p_N)$, with p_i either a probability or a sub-probability distribution over $\Sigma_i = \{a_{i-1}, a_i\}$, such that the random trace $\xi \in \overline{\mathcal{M}}$ produced by the PSA *based on $(p_1, \ldots, p_N)$ is finite with probability 1 and has the distribution $\mathbb{P}$.*

Proof. Using the same notations as in the proof of Theorem 5.3, we define $(p_i)_{1 \leq i \leq N}$ as follows:

$$p_1(a_0) = t_0 \qquad p_1(a_1) = 1 - t_0 \tag{24}$$

$$1 < i < N \quad p_i(a_{i-1}) = t_{i-1}\frac{\mu_{0,i-3}}{\mu_{0,i-2}} \qquad p_i(a_i) = \frac{\mu_{0,i-1}}{\mu_{0,i-2}} \tag{25}$$

$$p_N(a_{N-1}) = \frac{t_{N-1}}{p_{N-1}(a_{N-1})} \qquad p_N(a_N) = t_N \tag{26}$$

The only difference with the definitions introduced in the proof of Theorem 5.3 lies in (26). As a consequence, all $p_i(a_{i-1})$ and $p_i(a_i)$ are positive. They satisfy $p_i(a_i)p_{i+1}(a_i) = t_i$ for all $i \in \{2, \ldots, N-1\}$, and obviously $p_1(a_0) = t_0$ and $p_N(a_N) = t_N$. What remains to be proved is that the sums $p_i(a_{i-1}) + p_i(a_i)$ stay within $(0, 1]$ for all $i \in \{1, \ldots, N\}$.

For $i \in \{1, \ldots, N-1\}$, the sum is 1, just as in the proof of Theorem 5.3. And for $i = N$, we have:

$$\begin{aligned} p_N(a_{N-1}) + p_N(a_N) \leq 1 &\iff t_N + t_{N-1}\frac{\mu_{0,N-3}}{\mu_{0,N-2}} \leq 1 \\ &\iff \mu_{0,N-2} - t_N\mu_{0,N-2} - t_{N-1}\mu_{0,N-3} \geq 0 \\ &\iff \mu_{\mathcal{M}}(t_0, \ldots, t_N) \geq 0 \end{aligned}$$

The last condition is satisfied since $\mathbb{P}$ is a sub-Bernoulli measure. The proof is complete. □

5.4 — Finitary cases

5.4.1 *Example of a ring model* — Consider the ring model already introduced in Section 2.2. The trace monoid is:

$$\mathcal{M} = \langle a_0, a_1, a_2, a_3 \mid a_0a_2 = a_2a_0,\ a_1a_3 = a_3a_1 \rangle.$$

This is the synchronization monoid associated with the network of alphabets $(\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)$ given by $\Sigma_0 = \{a_3, a_0\}$, $\Sigma_1 = \{a_0, a_1\}$, $\Sigma_2 = \{a_1, a_2\}$ and $\Sigma_3 = \{a_2, a_3\}$. Contrasting with the path model, we shall see on an example that the PSA for this ring model with four generators produces finite traces with probability 1.

Let $p_0, \ldots, p_3$ be uniform distributions on $\Sigma_0, \ldots, \Sigma_3$. According to Theorem 5.1, the PSA yields a multiplicative and uniform probability measure $\mathbb{P}$ on $\overline{\mathcal{M}}$ with parameter $p = (1/2)(1/2) = 1/4$:

$$\forall x \in \mathcal{M} \quad \mathbb{P}(\Uparrow x) = \left(\frac{1}{4}\right)^{|x|}. \tag{27}$$

The Möbius polynomial of $\mathcal{M}$ is $\mu_{\mathcal{M}}(t) = 1 - 4t + 2t^2$, which root of smallest modulus is $p_0 = 1 - 1/\sqrt{2}$. Since $1/4 < p_0$, it follows from Theorem 4.4 that the PSA produces a finite trace $y \in \mathcal{M}$ with probability 1.

The average length of y is easily computed. Following the notations introduced in Section 4.2, we put $G(z) = \sum_{x \in \mathcal{M}} z^{|x|}$, $p = 1/4$ and $\varepsilon = \mu_{\mathcal{M}}(p)$. Using that $\mathbb{P}(y = x) = \varepsilon p^{|x|}$ and $G(z) = 1/\mu_{\mathcal{M}}(z)$, we have:

$$\mathbb{E}|y| = \sum_{x \in \mathcal{M}} |x| \mathbb{P}(y = x) = \varepsilon p \frac{dG}{dz}\Big|_{[z=p]} = -p \frac{\mu'_{\mathcal{M}}(p)}{\mu_{\mathcal{M}}(p)} = 6.$$

5.4.2 *Generalization: trees and cycles* — Since the probability distribution of the random element produced by the PSA is a multiplicative probability measure, we know by Theorem 4.3 that the trace is either finite with probability 1 or infinite with probability 1. The examples studied above show that both cases may occur indeed.

It turns out that the dichotomy is solved by a simple criterion, as stated below.

• Proposition 5.5—*Assume that the synchronization monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$ associated with a tuple of alphabets $(\Sigma_1, \ldots, \Sigma_N)$ is irreducible, and let D be the dependence relation of $\mathcal{M}$. Let $p_1, \ldots, p_N$ be positive probability distributions on $\Sigma_1, \ldots, \Sigma_N$ respectively. Then*

1. *The* PSA *based on $(p_1, \ldots, p_N)$ produces an infinite trace with probability* 1 *if and only if:*
   a) *$i \neq j \implies |\Sigma_i \cap \Sigma_j| \leq 1$; and*
   b) *the non oriented graph (Σ, D) has no cycle.*
2. *If the output X of the* PSA *is finite with probability* 1*, then the length $|X|$ has a finite average.*

Proof. Assume first that both conditions 1a and 1b are met. Let $Y = (Y_i)_{i \in \Sigma}$ be a tuple of infinite sequences $Y_i \in (\Sigma_i)^\omega$. With probability 1, for any distinct i and j such that $\Sigma_i \cap \Sigma_j \neq \emptyset$, the sequence Y_i has infinitely many occurrences of the unique element belonging to $\Sigma_i \cap \Sigma_j$. Just as in the case of the path model, this is enough to guarantee that the synchronization trace of Y is infinite.

Conversely, assume that one of conditions 1a and 1b is not met, for instance condition 1a. Let i and j be distinct indices such that $\Sigma_i \cap \Sigma_j$ has at least two distinct elements a and b. Let $Y = (Y_r)_{r \in \Sigma}$ be a random vector of sequences such that the synchronization trace of Y is infinite. Since $\mathcal{M}$ is assumed to be irreducible, we observe that, with probability 1, if the synchronization trace of Y is infinite then all coordinates of Y are infinite. Hence there is no loss of generality in assuming that both coordinate Y_i and Y_j are infinite. Then the order of occurrences of a and b in both coordinates must be the same, which has probability 0 to occur.

Finally, assume that condition 1b is not met, hence the presence of a cycle $(r_1, r_2, \ldots, r_k)$ in the dependence relation. We assume without loss of generality that $r_1, \ldots, r_{k-1}$ are pairwise disjoint. Let $a_1 \in \Sigma_{r_1} \cap \Sigma_{r_2}$, $a_2 \in \Sigma_{r_2} \cap \Sigma_{r_3}, \ldots,$ $a_k \in \sigma_{r_k} \cap \Sigma_{r_1}$. Focusing on the coordinates $r_1, \ldots, r_k$ of Y only, a pattern of the form $(a_1a_k, a_2a_1, a_3a_2, \ldots, a_{k-1}a_{k-2}, a_ka_{k-1})$ shall occur with probability 1. Since such a pattern is blocking the synchronization, it follows that the synchronization trace is finite with probability 1.

We have proved so far the equivalence stated in point 1. We now come to the proof of point 2, and assume that $|X| < \infty$ with probability 1. The average of $|X|$ is computed as the following mathematical expectation:

$$\mathbb{E}|X| = \sum_{x \in \mathcal{M}} |x| \mathbb{P}(X = x) = \varepsilon \sum_{x \in \mathcal{M} \setminus \{\mathbf{e}\}} |x| f(x),$$

where f is the sub-Möbius valuation associated with $\mathbb{P}$, and ε is the constant given by the Möbius transform of f evaluated at the empty heap (see Section 4.2.4). Let $G(\lambda)$ be the power series:

$$G(\lambda) = \sum_{x \in \mathcal{M}} \lambda^{|x|} f(x)$$

Then $G(1) = \varepsilon^{-1} < \infty$ according to Theorem 4.1 point 2, and since G has non negative coefficients, it implies by the Pringsheim Theorem that the radius of convergence of G is greater than 1. Hence so does its derivative, and thus $G'(1) < \infty$. Since $\mathbb{E}|X| = \varepsilon G'(1)$, the result of point 2 follows. □

6—The Probabilistic Full Synchronization Algorithm

The PSA produces random traces, either finite or infinite. Proposition 5.5 shows that the ability of the PSA to produce finite or infinite traces does not depend on the probabilistic parameters one chooses to equip the local alphabets with. It rather depends on the structure of the synchronization monoid.

To be sound, testing procedures and statistical averaging techniques require arbitrary large traces, which binds us to the mathematical model of infinite traces. In case where the PSA fails to produce infinite traces, we are thus left with an unsolved problem. Yet, we can produce finite traces... and the most natural thing to try from there, is to start the PSA over and over, and to concatenate the finite random traces obtained at each execution of the PSA. The limiting trace is infinite and random, couldn't it just be the one we were looking for?

It is quite surprising to realize that this strategy fails in general. Of course the unlimited concatenation of finite traces, with a positive average length, necessarily produces an infinite trace. But the failure comes from the *distribution* of this random infinite trace. We will show on an example below that it is *not* a Bernoulli measure in general. In particular, the uniform measure is thus unreachable by this technique.

Nevertheless, we introduce an algorithm based on recursive concatenation of finite random traces and that outputs, if executed indefinitely, an infinite trace which is always distributed according to a Bernoulli measure. The intermediate, finite random traces, are obtained by a trial-and-reject procedure based on the PSA. The whole procedure constitutes the Probabilistic Full Synchronization Algorithm (PFSA). In Section 6.4, we show that any Bernoulli measure can be simulated by the output of the PFSA for the ring model.

6.1 — Convolution of probability distributions and random walks

6.1.1 *Convolution and random walks* — We recall the general definition of convolution for a countable monoid $\mathcal{M}$. Let ν and θ be two probability distributions on $\mathcal{M}$. Assume that X and Y are two independent random variables with values in $\mathcal{M}$, distributed according to ν and to θ respectively. Then the *convolution* $\nu * \theta$ is the distribution of the random variable $X \cdot Y$, and it is given by the Cauchy product formula:

$$\forall x \in \mathcal{M} \quad \nu * \theta(x) = \sum_{(y,z)\in\mathcal{M} \,:\, y\cdot z = x} \nu(y) \cdot \theta(z).$$

The convolution product is associative.

Given a probability distribution ν over $\mathcal{M}$, let $(X_n)_{n\geq 1}$ be a sequence of independent random variables with values in $\mathcal{M}$, and identically distributed according to ν. The *random walk* associated with ν is the sequence of random variables $(Y_n)_{n\geq 0}$ defined by $Y_0 = \mathbf{e}$, the unit element of the monoid, and inductively: $Y_{n+1} = Y_n \cdot X_{n+1}$ for all integers n. If ν_n denotes the distribution of Y_n, we have $\nu_n = \nu^{*(n)}$ for all integers n, the n^{th} convolution power of ν with itself.

Each trajectory $(Y_n)_{n\geq 0}$ of the random walk is nondecreasing for the divisibility relation in the monoid. Hence, if we assume now that $\mathcal{M}$ is a trace monoid, the nondecreasing sequence $(Y_n)_{n\geq 1}$ has a least upper bound in the completion $\overline{\mathcal{M}}$, say $Y_\infty = \bigvee_{n\geq 1} Y_n$. We introduce the notation $\nu^{*\infty}$ for the probability distribution of Y_∞, which we call the *limit distribution of the random walk* (it is also called the harmonic measure of the random walk).

6.1.2 *An example where the concatenation of* PSA *traces does not yield a limit multiplicative measure* — We consider the ring synchronization monoid on four generators $\mathcal{M} = \langle a_0, a_1, a_2, a_3 \mid a_0a_2 = a_2a_0,\ a_1a_3 = a_3a_1\rangle$, corresponding to the network of alphabets $(\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)$ with $\Sigma_0 = \{a_3, a_0\}$, $\Sigma_1 = \{a_0, a_1\}$, $\Sigma_2 = \{a_1, a_2\}$ and $\Sigma_3 = \{a_2, a_3\}$. Each alphabet Σ_i, for $i = 0, \ldots, 3$, is equipped with the uniform probability distribution. Let $\nu^{*\infty}$ be the distribution on $\partial\mathcal{M}$ of the infinite trace obtained by concatenating infinitely many independent copies of a finite trace generated by the PSA. Then we claim: *the limit distribution $\nu^{*\infty}$ is not Bernoulli.*

Seeking a contradiction, assume that it is. Clearly, $\nu^{*\infty}$ is concentrated on the boundary $\partial\mathcal{M}$. And for symmetry reasons, it is necessarily the uniform distribution, and thus given by $\nu^{*\infty}(\uparrow x) = p_0^{|x|}$ where p_0 is the root of smallest modulus of the Möbius polynomial $\mu(z) = 1 - 4z + 2z^2$.

We extend the concatenation of traces $x\cdot y$ with $(x, y) \in \mathcal{M}\times\mathcal{M}$ to the case where y is an infinite trace by putting $x\cdot y = \bigvee\{x\cdot y_n \ :\ n \geq 1\}$ for $y = \bigvee\{y_n \ :\ n \geq 1\}$, and this definition does not depend on the choice of the sequence $(y_n)_{n\geq 1}$. This yields also an extension of the notion of convolution $\nu * \theta$ to the case where ν is concentrated on $\mathcal{M}$, but θ might be a probability distribution on $\overline{\mathcal{M}}$. The construction of $\nu^{*\infty}$ implies the fix point property $\nu_p * \nu^{*\infty} = \nu^{*\infty}$, where $\nu_p(\Uparrow x) = p^{|x|}$ is the distribution of the PSA, here given by $p = 1/4$. In particular for the cylinder $\uparrow a_0$, this yields:

$$p_0 = \sum_{k\geq 0} \nu_p(\{a_2^k\})\nu^{*\infty}(\uparrow a_0) + \nu_p(\Uparrow a_0) = p_0(1 - 4p + 2p^2)\frac{1}{1-p} + p$$

Simplifying by $p \neq 0$, we obtain: $p = (3p_0-1)/(2p_0-1)$, and since $1-4p_0+2p_0^2 = 0$, it yields $p = p_0$, a contradiction. Actually, we have shown the strongest result that no random walk based on the distributions $\nu_p(\Uparrow x) = p^{|x|}$ with $p \in (0, p_0)$ has a Bernoulli measure as limit distribution $\nu_p^{*\infty}$.

6.2 — First hitting times and pyramidal heaps

6.2.1 *First hitting times* — First hitting times for random heaps formalize the idea of the first time of occurrence of a given piece—yet, without an explicit notion of time at hand. It generalizes to random heaps the analogous notion, for a Bernoulli sequence, of first time of reaching a given letter.

Let $\mathcal{M} = \mathcal{M}(\Sigma, I)$ be a trace monoid, and let $a \in \Sigma$ be a given letter. The number of occurrences of a in the congruent words defining a trace x is constant, and depends thus only on the trace x. We denote it $|x|_a$. For any infinite trace $\xi \in \partial\mathcal{M}$, let $L_a(\xi) = \{x \in \mathcal{M} \,:\, x \leq \xi \wedge |x|_a > 0\}$. If non empty, the set $L_a(\xi)$ has a minimum which we denote by $V_a(\xi)$, and it satisfies $|V_a(\xi)|_a = 1$. Intuitively, $V_a(\xi)$ represents the smallest sub-trace of ξ with at least an occurrence of a.

If $\partial\mathcal{M}$ is equipped with a Bernoulli measure $\mathbb{P}$, then $L_a(\xi) \neq \emptyset$ with probability 1. Hence, neglecting a set of zero probability, we may assume that $V_a : \partial\mathcal{M} \to \mathcal{M}$ is well defined. The mapping V_a is called the *first hitting time of* a. The *distribution of the first hitting time of* a is the probability distribution of the random variable V_a. It is a discrete probability distribution on $\mathcal{M}$, which we denote by $\mathbb{P}_a$, and which is defined by $\mathbb{P}_a(x) = \mathbb{P}(V_a = x)$ for all $x \in \mathcal{M}$.

We will base our random generation of infinite heaps on the following result.

• Theorem 6.1—*Let $\mathbb{P}$ be a Bernoulli measure equipping the boundary $\partial\mathcal{M}$ of an irreducible trace monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$. Let $a \in \Sigma$, and let $\mathbb{P}_a$ be the distribution of the first hitting time of a. Then $\mathbb{P}_a^{*\infty} = \mathbb{P}$, where $\mathbb{P}_a^{*\infty}$ is the limit distribution of the random walk on $\mathcal{M}$ associated with $\mathbb{P}_a$.*

Sketch of proof. Let $(V^n)_{n\geq 0}$ be the sequence of iterated stopping times associated with the first hitting time V_a, as defined in [1, Def. 5.2]. Under the probability $\mathbb{P}$, it follows from [1, Prop. 5.3] that, for each integer n, V^n has the same distribution as the n^{th} step of the random walk associated with $\mathbb{P}_a$. Since $\mathcal{M}$ is assumed to be irreducible, the sequence $(V^n)_{n\geq 0}$ is exhaustive as defined in [1, Def. 5.5], from which the result derives. □

As a consequence, if we can simulate the distribution $\mathbb{P}_a$ of the first hitting time of some piece a, we will be able to simulate a $\mathbb{P}$-distributed infinite random heap. The improvement lies in the fact that first hitting times are finite heaps. Our next task consists thus in studying more closely the distribution of the first hitting time, after which we shall see how to simulate it.

6.2.2 *Pyramidal heaps and the distribution of the first hitting time* — Recall that any trace has a interpretation as a labeled partial order of pieces (see Section 3.2.2). A trace x is *pyramidal* if, as labeled partially ordered set, it has a unique maximal element (a notion introduced by Viennot [22]). Any trace of the form $x = V_a(\xi)$ for some $\xi \in \partial\mathcal{M}$ is pyramidal, with its unique occurrence of a as its unique maximal piece.

Then the set $\mathcal{V}_a$ of traces $x \in \mathcal{M}$ in the image of the mapping $V_a : \partial\mathcal{M} \to \mathcal{M}$ can be described as follows: $\mathcal{V}_a$ is the set of pyramidal traces $x \in \mathcal{M}$ such that the piece a only occurs as the unique maximal piece; see Figure 7. Furthermore, we observe that, if $x \in \mathcal{V}_a$, then $\{V_a = x\} = \uparrow x$ (an intuitive property, also proved in [1, Prop. 4.2]). It follows that the distribution $\mathbb{P}_a$ of the first hitting time has the following simple expression:

$$\forall x \in \mathcal{M} \quad \mathbb{P}_a(x) = \begin{cases} 0, & \text{if } x \notin \mathcal{V}_a \\ \mathbb{P}(\uparrow x), & \text{if } x \in \mathcal{V}_a \end{cases} \tag{28}$$

6.2.3 *Generating pyramidal heaps* — We consider a network of alphabets $(\Sigma_1, \ldots, \Sigma_N)$ with $\Sigma = \Sigma_1 \cup \ldots \Sigma_N$, such that the synchronization trace monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$ is irreducible. We pick an arbitrary letter $a \in \Sigma$. For each $i \in \{1, \ldots, N\}$, let

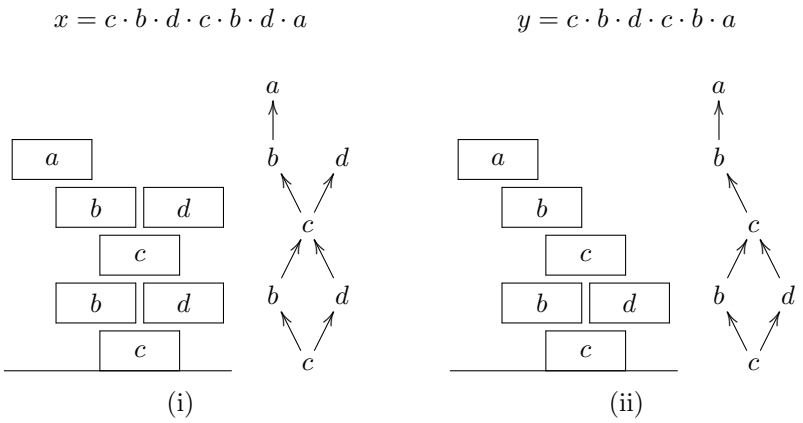


Figure 7: (i): *For* $\mathcal{M} = \langle a, b, c, d \ : \ ac = ca, \ ad = da, \ bd = db \rangle$*: a trace* $x \notin \mathcal{V}_a$ *(it is not pyramidal) and its associated labeled poset.* (ii)*: a trace* $y \in \mathcal{V}_a$ *and its associated labeled poset on the right.*

$\Sigma'_i = \Sigma_i \setminus \{a\}$ if $a \in \Sigma_i$, and $\Sigma'_i = \Sigma_i$ otherwise. Then the synchronization monoid $\mathcal{M}'$ of $(\Sigma'_1, \ldots, \Sigma'_N)$ coincides with the sub-monoid of $\mathcal{M}$ generated by $\Sigma \setminus \{a\}$.

We assume, for each $i \in \{1, \ldots, N\}$, to be given p'_i, either a probability distribution or a sub-probability distribution over Σ'_i, such that the output X of the PSA executed on $(\Sigma'_1, \ldots, \Sigma'_N)$ with those parameters, is finite with probability 1.

Given the parameters $(p'_1, \ldots, p'_N)$, we consider the execution of the trial-and-reject procedure described in pseudo-code in Algorithm 5 below. At each run of the loop, the algorithm needs to decide whether some trace V is pyramidal in $\mathcal{M}$ or not. It is clear that a—far from being optimal—scanning procedure, examining all elements of V starting from the right, will successfully complete this job.

Algorithm 5 Outputting a pyramidal trace in $\mathcal{V}_a$

Require: –

$X \leftarrow \mathbf{e}$ ▷ Initialization
repeat
  call Algorithm 4 ▷ Calling the PSA
  $X \leftarrow$ output of Algorithm 4
  $V \leftarrow X \cdot a$
until V is pyramidal in $\mathcal{M}$ ▷ See comment
return V

By assumption, the PSA which is repeatedly called in Algorithm 5 always terminates and outputs a finite trace. Since the probability of success if positive, since for instance $X = \mathbf{e}$ yields a success and has positive probability to be output by the PSA, Algorithm 5 always terminates. The probability distribution of the output is given in the following lemma, stated with the notations introduced above.

• Lemma 6.2—*Let* $f' : \mathcal{M}' \to \mathbb{R}_+^*$ *be the sub-Möbius valuation associated with the output of the* PSA *running on* $\mathcal{M}'$*. Then the distribution of the output* V *of Algorithm 5 is concentrated on* $\mathcal{V}_a$ *and given by:*

$$\forall v \in \mathcal{V}_a \quad \mathbb{P}(V = v) = K \cdot f'(v/a), \tag{29}$$

where v/a denotes the heap obtained by removing from v its unique maximal piece a, and K is a normalization constant.

Proof. Let $\mathbb{Q}$ denote the probability distribution of the output X of the PSA running on $\mathcal{M}'$ with the specified parameters $(p'_1, \dots, p'_N)$. Then $f'(x) = \mathbb{Q}(\Uparrow x)$ for $x \in \mathcal{M}'$. According to Theorem 4.3, for some constant $\varepsilon > 0$, we have $\mathbb{Q}(X = x) = \varepsilon f'(x)$ for all $x \in \mathcal{M}'$.

The rejection procedure amounts to considering the distribution of $V = X \cdot a$ conditioned on $X \cdot a \in \mathcal{V}_a$. Hence the probability for Algorithm 5 to issuing an element $v \in \mathcal{V}_a$ is:

$$\mathbb{P}(V = v) = \frac{\mathbb{Q}(X \cdot a = v)}{\mathbb{Q}(X \cdot a \in \mathcal{V}_a)} = \frac{\mathbb{Q}(X = v/a)}{\mathbb{Q}(X \cdot a \in \mathcal{V}_a)} = K \cdot f'(v/a),$$

where K is the constant $K = \varepsilon/\mathbb{Q}(X \cdot a \in \mathcal{V}_a)$. □

Note that the form (29) is almost that of a valuation evaluated at v. The contribution of the last piece a is missing, but the constant K is adequately placed to play the role of this missing contribution. This will be used in the proof of Theorem 6.3.

6.3 — The Probabilistic Full Synchronization Algorithm

We are now ready for constructing a probabilistic algorithm generating Bernoulli-distributed infinite traces. The framework consists of a network $(\Sigma_1, \dots, \Sigma_N)$ of alphabets, such that the synchronization trace monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$ is irreducible, with $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_N$.

6.3.1 *Description of the algorithm* — Having chosen an arbitrary piece $a \in \Sigma$, we consider a family $(p'_1, \dots, p'_N)$ of probabilistic parameters as above, *i.e.*, with the constraint that the PSA executed on the sub-monoid $\mathcal{M}'$ generated by $\Sigma \setminus \{a\}$ and with these parameters, outputs a finite trace with probability 1.

The Probabilistic Full Synchronization Algorithm (PFSA) is described in pseudo-code in Algorithm 6 below. The PFSA is an endless loop, incrementally writing to its output register X. It simulates thus the random walk on $\mathcal{M}$ with increments distributed according to the distribution established in Lemma 6.2.

Algorithm 6 Probabilistic Full Synchronization Algorithm

Require: –

```
X ← e                                          ▷ Initialization
repeat
    call Algorithm 5
    V ← output of Algorithm 5                  ▷ Random pyramidal trace V ∈ V_a
    X ← X · V                                  ▷ Increments the random walk
until false
```

The analysis of Algorithm 6 is twofold: a probabilistic analysis carried on below and a complexity analysis carried on in Section 6.3.3.

6.3.2 *Probabilistic analysis of the* PFSA — Recall that, by convention, the *output* of the PFSA is the random infinite heap, least upper bound in $\partial\mathcal{M}$ of the finite heaps recursively written in its output register. The probability distribution of this infinite heap is as follows.

• Theorem 6.3— *We consider the execution of the* PFSA *in the framework described in Section 6.3.1, and we adopt the same notations. Let $f' : \mathcal{M}' \to \mathbb{R}_+^*$ be the sub-Möbius valuation associated with the* PSA *executed with the chosen parameters $(p'_1, \dots, p'_N)$, and let X_∞ be the output (with the convention recalled above) of the* PFSA.

Then X_∞ is distributed according to a Bernoulli measure on $\partial\mathcal{M}$. The associated valuation $f : \mathcal{M} \to \mathbb{R}$ is the Möbius valuation on $\mathcal{M}$ that extends $f' : \mathcal{M}' \to \mathbb{R}$ (the existence and uniqueness of which are stated in Theorem 4.5).

Proof. Let $f : \mathcal{M} \to \mathbb{R}_+^*$ be as in the statement. Let $\mathbb{P}$ be the Bernoulli measure on $\partial\mathcal{M}$ defined by $\mathbb{P}(\uparrow x) = f(x)$, which is well defined according to Theorem 4.3.

Letting $\mathbb{Q}$ be the distribution of X_∞, we have to prove that $\mathbb{P} = \mathbb{Q}$. Let $g : \mathcal{V}_a \to \mathbb{R}_+^*$ be the probability distribution of the increment V in the PFSA. Since $\mathbb{Q}$ is the limit distribution of the random walk on $\mathcal{M}$ with increments distributed identically to V, it follows from Theorem 6.1 that we only need to show that $g(x) = \mathbb{P}(V_a = x)$ for all $x \in \mathcal{V}_a$.

According to (28), we have:

$$\forall x \in \mathcal{V}_a \quad \mathbb{P}(V_a = x) = \mathbb{P}(\uparrow x) = f(x). \tag{30}$$

Whereas, according to Lemma 6.2, we have for some constant K:

$$\forall x \in \mathcal{V}_a \quad g(x) = Kf'(x/a) = \frac{K}{f(a)} f(x). \tag{31}$$

Summing up over $\mathcal{V}_a$ in (30) yields $\sum_{x\in\mathcal{V}_a} f(x) = 1$, whereas summing up over $\mathcal{V}_a$ in (31) yields $1 = \big(K/f(a)\big) \cdot \big(\sum_{x\in\mathcal{V}_a} f(x)\big) = K/f(a)$. Therefore $K = f(a)$, which yields after re-injecting in (31): $g(x) = f(x)$ and thus $g(x) = \mathbb{P}(V_a = x)$ for all $x \in \mathcal{V}_a$. Since $\mathbb{P}$ has the desired properties, the proof is complete. □

6.3.3 *Complexity analysis of the* PFSA — We will limit our analysis to the following observation: *the size of the output register of the* PFSA *grows linearly with time in average.*

Indeed, each PSA call in Algorithm 5 takes a finite amount of time in average according to Proposition 5.5, point 2. Since the probability of success in the trial-and-reject procedure of Algorithm 5 is positive, it will thus execute in average in a fixed amount of time, whence the average linear growth of the output register of the PFSA.

Another question is to compute adequately the probabilistic parameters. We will discuss it briefly in Section 7, after having examined some examples.

6.4 — Example: ring models

6.4.1 *A general result* — For the ring models, the following result shows that any Bernoulli measure can be simulated by executions of the PFSA.

• Theorem 6.4—*Let $(a_0, \ldots, a_{N-1})$ be N distinct symbols, and let $(\Sigma_0, \ldots, \Sigma_{N-1})$ be the network of alphabets defined by:*

$$0 < i \leq N-1 \quad \Sigma_i = \{a_{i-1}, a_i\}, \qquad \Sigma_0 = \{a_{N-1}, a_0\}.$$

The synchronization monoid $\mathcal{M} = \mathcal{M}(\Sigma, I)$ is described as follows:

$$\Sigma = \{a_0, \ldots, a_{N-1}\}, \quad I = \{(a_i, a_j) \ : \ (i - j \mod N \geq 2) \wedge (j - i \mod N \geq 2)\}.$$

Then any Bernoulli measure on $\partial\mathcal{M}$ can be simulated by the endless execution of the PFSA*, and in particular the uniform measure on $\partial\mathcal{M}$.*

Proof. Let $\mathbb{P}$ be a target Bernoulli measure on $\partial\mathcal{M}$, and let $f : \mathcal{M} \to \mathbb{R}_+^*$ be the associated Möbius valuation. We pick a_0 as the piece to be removed. Let $\mathcal{M}'$ be the submonoid of $\mathcal{M}$ generated by $a_1, \ldots, a_N$. Then $\mathcal{M}'$ is a path model. Furthermore,

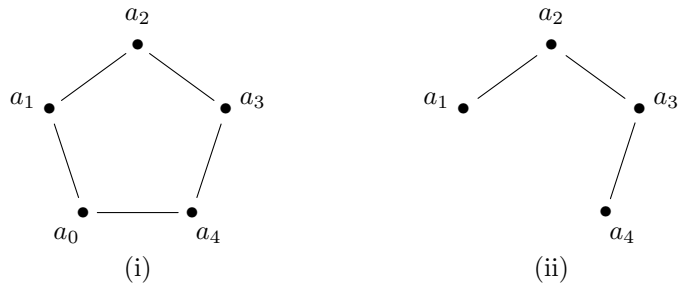


Figure 8: (i): *synchronization monoid for the ring model with five generators.* (ii): *generators of the path model on which the* PSA *is run at each loop of the* PFSA.

since $\mathcal{M}$ is irreducible, it follows from Theorem 4.5 point 1 that the restriction of f to $\mathcal{M}'$ is sub-Möbius. According to Theorem 5.4, the associated probability distribution can be obtained by running the PSA with suitable parameters. Running the PFSA based on this instance of the PSA, we generate a Bernoulli measure which Möbius valuation, say g, extends to $\mathcal{M}$ the restriction of f to $\mathcal{M}'$. By the uniqueness of the extension of sub-Möbius valuations to Möbius valuations (Theorem 4.5 point 2), it follows that $g = f$. □

6.4.2 *Example* — We consider the example of the ring model $\mathcal{M}$ on five generators, the synchronization graph of which is depicted in Figure 8, (i). We aim at generating the uniform measure on $\partial\mathcal{M}$, say ν, characterized by $\nu(\uparrow x) = p_0^{|x|}$, where $p_0 = \frac{1}{2} - \frac{\sqrt{5}}{10}$ is the root of smallest modulus of $\mu_{\mathcal{M}}(t) = 1 - 5t + 5t^2$.

We pick a_0 as our distinguished piece, as depicted in Figure 8, (ii). Then, we wish to run on the sub-monoid

$$\mathcal{M}' = \langle a_1, a_2, a_3, a_4 \mid a_1a_3 = a_3a_1,\ a_1a_4 = a_4a_1,\ a_2a_4 = a_4a_2 \rangle$$

a PSA with associated valuation $f' : \mathcal{M}' \to \mathbb{R}$ such that $f'(x) = p_0^{|x|}$ for all $x \in \mathcal{M}'$. Note that p_0 is *not* the root of $\mu_{\mathcal{M}'}$!

Referring to the computations performed in the proof of Theorem 5.4, adequate solutions for p_2, p_3 and p_4, respectively on $\{a_1, a_2\}$, on $\{a_2, a_3\}$ and on $\{a_3, a_4\}$, are obtained as follows:

$$p_2(a_1) = p_0 = \frac{1}{2} - \frac{\sqrt{5}}{10} \approx 0.276 \qquad p_2(a_2) = 1 - p_0 = \frac{1}{2} + \frac{\sqrt{5}}{10} \approx 0.724$$

$$p_3(a_2) = \frac{p_0}{1 - p_0} = \frac{3}{2} - \frac{\sqrt{5}}{2} \approx 0.382 \qquad p_3(a_3) = \frac{1 - 2p_0}{1 - p_0} = -\frac{1}{2} + \frac{\sqrt{5}}{2} \approx 0.618$$

$$p_4(a_3) = \frac{p_0(1 - p_0)}{1 - 2p_0} = \frac{1}{\sqrt{5}} \approx 0.447 \qquad p_4(a_4) = p_0 = \frac{1}{2} - \frac{\sqrt{5}}{10} \approx 0.276$$

7—Computational issues and perspectives

When trying to use the PFSA in practice for simulation and testing, one might be concerned by the fact that it incrementally outputs heaps with a particular shape, namely they are all pyramidal. Furthermore, the tip of these pyramidal heaps is always labeled with the same letter, corresponding to an arbitrary choice made before executing the algorithm. Actually, there are good reasons not to worry about that. Indeed, a large class of statistics on heaps can safely be computed on these particular

heaps, and they will asymptotically be indistinguishable from statistics computed on arbitrary large heaps. For instance, an asymptotics of the speedup, *i.e.* the ratio height over number of pieces in large heaps, can be estimated in this way. Precise results on this topic are found in [1], under the name of cut-invariance.

Another concern is the following. Even if the PFSA were proved to be able to simulate any Bernoulli measure for any topology, not only for the ring topology, there is no doubt that its execution would still need the precomputation of adequate probabilistic parameters, and in particular the root of smallest modulus of the Möbius polynomial of the synchronization monoid. Given that the determination of this polynomial on a general synchronization graph is an NP-complete problem (since the independence set decision problem is NP-complete [8]), this precomputation method appears unrealistic to be used in practice.

However, what we need is not the Möbius polynomial itself, but only its root of smallest modulus. It is not theoretically forbidden to think that this root might be approximated in polynomial time, even though the Möbius polynomial is hard to find. Actually, one can even think of a feedback procedure based on our generation algorithms to find an approximation of this root. Indeed, we could execute the generation algorithm with arbitrary parameters, then adjust the probabilistic parameters in order to increase uniformity, then re-run the generation algorithm and re-adjust the parameters, ans so on. It is reasonable to expect that such a procedure would lead the parameters to converge toward the critical value entailing uniformity, which is the root of smallest modulus of the Möbius polynomial. This interesting question may deserve a dedicated work.

Acknowledgments. A number of ideas in this paper have emerged through animated discussions with É. Fabre, B. Genest and N. Bertrand at INRIA Rennes during Spring 2015. I am grateful to A. Muscholl for pointing out the reference [7]; and I am grateful to C. Male for pointing out the reference [15]. I am grateful to the anonymous reviewers who greatly helped me improving the paper.

References

[1] S. Abbes. "A cut-invariant law of large numbers for random heaps". In: *J. Theoret. Probab.* (2016), pp. 1–34. DOI: doi:10.1007/s10959-016-0692-6.

[2] S. Abbes and J. Mairesse. "Uniform and Bernoulli measures on the boundary of trace monoids". In: *J. Combin. Theory Ser. A* 135 (2015), pp. 201–236.

[3] S. Abbes and J. Mairesse. "Uniform generation in trace monoids". In: *Math. Found. Comput. Sc. 2015 (MFCS 2015), part 1*. Ed. by G. Italiano, G. Pighizzini, and D. Sannella. Vol. 9234. Lecture Notes in Comput. Sci. Springer, 2015, pp. 63–75.

[4] O. Bernardi and O. Giménez. "A linear algorithm for the random sampling from regular languages". In: *Algorithmica* 62.1–2 (2012), pp. 130–145.

[5] P. Billingsley. *Probability and Measure, 3rd edition*. Wiley, 1995.

[6] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements*. Vol. 85. Lecture Notes Math. Springer, 1969.

[7] R. Cori and D. Perrin. "Automates et commutations partielles". In: *RAIRO Theor. Inform. Appl.* 19.1 (1985), pp. 21–32.

[8] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill, 2006.

[9] A. Denise et al. “Uniform random sampling of traces in very large models”. In: *Proc. first internat. workshop on Random Testing (RT'06)*. ACM Press, 2006, pp. 10–19.

[10] V. Diekert. *Combinatorics on Traces*. Vol. 454. Lecture Notes Comput. Sci. Springer, 1990.

[11] V. Diekert and G. Rozenberg, eds. *The Book of Traces*. World Scientific, 1995.

[12] P. Duchon et al. “Boltzmann samplers for the random generation of combinatorial structures”. In: *Combin. Probab. Comput.* 13 (2004), pp. 577–625.

[13] B. Genest et al. “Asynchronous games over tree architectures”. In: *Proc. 40th Internat. Colloquium on Automata, Languages and Programming (ICALP 2013)*. Vol. 7966. Lecture Notes Comput. Sci. Springer, 2013, pp. 275–286.

[14] M. Goldwurm and M. Santini. “Clique polynomials have a unique root of smallest modulus”. In: *Inform. Process. Lett.* 75.3 (2000), pp. 127–132.

[15] K. Khanin et al. “Ballistic deposition patterns beneath a growing Kardar-Parisi-Zhang interface”. In: *Phys. Rev. E* 82 (2010).

[16] D. Krob, J. Mairesse, and I. Michos. “Computing the average parallelism in trace monoids”. In: *Discrete Math.* 273 (2003), pp. 131–162.

[17] A. Mazurkiewicz. “Trace theory”. In: *Petri nets: applications and relationships to other models of concurrency*. Vol. 255. Lecture Notes Comput. Sci. Springer, 1987, pp. 278–324.

[18] J. Oudinet. “Uniform random walks in very large models”. In: *Proc. second internat. workshop on Random Testing (RT'07)*. ACM Press, 2007, pp. 26–29.

[19] W. Reisig and G. Rozenberg, eds. *Lectures on Petri nets I: basic models*. Vol. 1491. Lecture Notes Comput. Sci. Springer, 1998.

[20] K. Sen. “Effective random testing of concurrent programs”. In: *Proc. twenty-second IEEE/ACM internat. conference on Automated software engineering (ASE'07)*. ACM Press, 2007, pp. 323–332.

[21] E. Seneta. *Non-negative Matrices and Markov Chains. Revised printing*. Springer, 1981.

[22] X. Viennot. “Heaps of pieces, I : basic definitions and combinatorial lemmas”. In: *Combinatoire énumérative*. Vol. 1234. Lecture Notes Math. Springer, 1986, pp. 321–350.

[23] X. Viennot. “Problèmes combinatoires posés par la physique statistique. Exposé n° 626”. In: *Séminaire N. Bourbaki, 1983–1984*. Astérisque. Soc. Math. France, 1985, pp. 225–246.